

2014 IEEE 22nd International Requirements Engineering Conference (RE)

Proceedings

Tony Gorschek (General Chair) and
Robyn Lutz (PC Chair)

August 25-29, 2014
Karlskrona, Sweden

2014 IEEE 22nd International Requirements Engineering Conference (RE)

IEEE Catalog Number: CFP14022-PRT

ISBN: 978-1-4799-3031-9

Copyright and Reprint Permission:

Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limit of U.S. copyright law for private use of patrons those articles in this volume that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923. For other copying, reprint or republication permission, write to IEEE Copyrights Manager, IEEE Operations Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331.

All rights reserved.

Copyright © 2014 by the Institute of Electrical and Electronics Engineers

Production: Conference Publishing Consulting, D-94034 Passau, Germany
info@conference-publishing.com

Message from the Chairs



Robyn Lutz
Program Chair



Tony Gorscok
General Chair



Sarah Gregory
Industry Co-Chair



Marjo Kauppinen
Industry Co-Chair

We warmly welcome you to RE'14, the 22nd IEEE International Requirements Engineering Conference! This year's conference takes place in Karlskrona, Sweden at the Blekinge Institute of Technology from August 25-29.

The IEEE International Requirements Engineering Conference is the premier international forum for researchers, industrial practitioners, educators, and students to present and discuss the most recent innovations, trends, experiences, results and challenges in the field of requirements engineering (RE).

The conference this year has a dual theme focused on innovation. Innovation *in* Requirements Engineering considers new ideas and breakthroughs in the area. Innovation *through* Requirements Engineering centers on how its practices, processes, models, methods and frameworks enable and support innovation in product development.

RE'14 features an extensive technical program of interest to both academia and industry. The research track addresses key problems and reports important new results in the field of requirements engineering from progress around the globe. The industry track offers experience reports and descriptions of challenges encountered in practice by requirements engineering professionals.

The three conference days are enhanced with panels, mini-tutorials, posters and tool demonstrations, and an Industry Lab. Panels give an opportunity for a mix of people to come together to offer their positions on particular RE topics of the day, with audience discussion encouraged. Mini-tutorials offered by experts give attendees an efficient introduction to emerging topics. Posters and tool demonstrations introduce solutions in an interactive setting, and are previewed this year with a fast-paced, lightning-round session. Industry Laboratory is a new event introduced this year to attract and involve industry in the applied field of requirements engineering research by enabling industrial participants to discuss experiences and share their visions for the future with the research community.

We are pleased to have three superb keynote speakers. Anthony I. Wasserman, Carnegie Mellon University-Silicon Valley, analyzes approaches used by tech startups to define product requirements. Annie I. Antón, Georgia Tech College of Computing, describes her work on requirements for software systems that must comply with privacy and security regulations. Anthony Finkelstein, University College London, discusses current issues in requirements engineering, technology and education.

The two days before the main conference feature a full range of offerings. Tutorials by experts give participants a popular way to develop new skills in requirements engineering practices. Several tutorials have been selected to be of particular interest to industrial practitioners. Multiple workshops encourage the exchange of ideas and discuss challenging research issues in requirements engineering. The workshops gather together participants with shared interests for a day of talks, discussion and activities. Several of these workshops are regularly held at the RE conference and return this year. The RE'14 Doctoral Symposium brings together PhD students and academics working in all areas of requirements engineering. Students present their research projects and receive feedback from a panel of internationally renowned researchers in a highly interactive and collegial format.

The research track received 115 full papers by authors from 30 countries. Of these, 77 were submitted as technical solution papers, 29 as scientific evaluation papers, and 9 as vision papers. The industry track received 35 papers by authors from 20 countries. In the first phase, three members of the Program Committee reviewed each research paper, and three members of the Industry Committee reviewed each industry paper. A discussion period among the reviewers followed, facilitated by a Program Board member. At the end of the discussion phase 59 research papers advanced to the Program Board meeting. The reviews and discussions were then used by the Program Board to make final selections. A total of 31 full research papers (25 technical solution and 6 scientific evaluation) were accepted for publication, an acceptance rate of 27%. 13 full industry papers were also accepted for publication, an acceptance rate of 37%. It is the high quality of these authors' papers that provides the strong technical program for RE'14.

The members of the RE'14 Organizing Committee, Program Committee, Industry Committee and Program Board have worked hard to provide you a high-quality, engaging and useful experience. We thank them all for their dedication and expertise. We also thank our corporate and academic donors for their generous support. Finally, we thank you for participating in RE'14, and welcome you to Karlskrona!

Robyn Lutz, RE'14 Program Chair
Tony Gorschek, RE'14 General Chair
Sarah Gregory, RE'14 Industry Co-Chair
Marjo Kauppinen, RE'14 Industry Co-Chair

Organization Committee

General Chair

Tony Gorschek

BLEKINGE INSTITUTE OF TECHNOLOGY, SWEDEN

Program Chair

Robyn Lutz

IOWA STATE UNIVERSITY, USA

Industry Track Co-Chair

Sarah C. Gregory

INTEL CORPORATION, USA

Industry Track Co-Chair

Marjo Kauppinen

AALTO UNIVERSITY, FINLAND

Organizing Chair

Jürgen Börstler

BLEKINGE INSTITUTE OF TECHNOLOGY, SWEDEN

Finance Chair

Samuel Fricker

BLEKINGE INSTITUTE OF TECHNOLOGY, SWEDEN

Industrial Coordination Chair

Robert Feldt

BLEKINGE INSTITUTE OF TECHNOLOGY, SWEDEN

Workshop Co-Chair

Travis Breaux

CARNEGIE MELLON UNIVERSITY, USA

Workshop Co-Chair

Eric Knauss

CHALMERS UNIVERSITY OF TECHNOLOGY AND GÖTEBORGS UNIVERSITET, SWEDEN

Tutorial Co-Chair

Erik Bjernulf

TOLPAGORNI PRODUCT MANAGEMENT AB, SWEDEN

Tutorial Co-Chair

Nan Niu

MISSISSIPPI STATE UNIVERSITY, USA

Posters, Demos and Exhibits Co-Chair
Jennifer Horkoff
UNIVERSITÀ DEGLI STUDI DI TRENTO, ITALY

Posters, Demos and Exhibits Co-Chair
Richard Berntsson Svensson
CHALMERS UNIVERSITY OF TECHNOLOGY AND GÖTEBORGS UNIVERSITET, SWEDEN

Doctoral Symposium Co-Chair
Mats Heimdahl
UNIVERSITY OF MINNESOTA, USA

Doctoral Symposium Co-Chair
Camille Salinesi
UNIVERSITÉ PARIS 1 PANTHÉON - SORBONNE, FRANCE

Industry Laboratory Chair
Alistair Mavin
ROLLS-ROYCE, UK

Industry Laboratory Deputy Chair
Eero Uusitalo
AALTO UNIVERSITY, FINLAND

Industry Laboratory Assistant
Cristina Palomares
UNIVERSITAT POLITÈCNICA DE CATALUNYA, SPAIN

Coordinating Publicity Chair
Gunter Mussbacher
MCGILL UNIVERSITY, CANADA

Industry Publicity Chair Sweden
Stefan Eekenulv
INCEPITIVE AB, SWEDEN

Supporting Publicity Chair Asia
Ye Yang
CHINESE ACADEMY OF SCIENCES, CHINA

Supporting Publicity Chair Europe
Norbert Seyff
UNIVERSITÄT ZÜRICH, SWITZERLAND

Supporting Publicity Chair North America
Birgit Penzenstadler
UNIVERSITY OF CALIFORNIA, USA

Supporting Publicity Chair South America
Emilia Mendes
BLEKINGE INSTITUTE OF TECHNOLOGY, SWEDEN

Supporting Publicity Chair Sweden
Mahvish Khurum
BLEKINGE INSTITUTE OF TECHNOLOGY, SWEDEN

Publication Chair
Kathryn Stolee
IOWA STATE UNIVERSITY, USA

Social Media Chair
Martin Mahaux
UNIVERSITÉ DE NAMUR, BELGIUM

Mobile Media Chair
Reid Holmes
UNIVERSITY OF WATERLOO, CANADA

Student Volunteer Chair
Krzysztof Wnuk
LUNDS UNIVERSITET, SWEDEN

Interaction Co-Chair
Mikael Svahnberg
BLEKINGE INSTITUTE OF TECHNOLOGY, SWEDEN

Web Chair, Interaction Co-Chair, Webmaster
Michael Unterkalmsteiner
BLEKINGE INSTITUTE OF TECHNOLOGY, SWEDEN

Program Board

Betty Cheng MICHIGAN STATE UNIVERSITY, USA
Anthony Finkelstein UNIVERSITY COLLEGE LONDON, UK
Xavier Franch UNIVERSITAT POLITÈCNICA DE CATALUNYA, SPAIN
Jane Cleland-Huang DEPAUL UNIVERSITY, USA
Zhi Jin PEKING UNIVERSITY, CHINA
Emmanuel Letier UNIVERSITY COLLEGE OF LONDON, UK
John Mylopoulos UNIVERSITY OF TORONTO, CANADA
Bashar Nuseibeh THE OPEN UNIVERSITY, UK
Tetsuo Tamai HOSEI UNIVERSITY, JAPAN
Axel van Lamsweerde UNIVERSITÉ CATHOLIQUE DE LOUVAIN, BELGIUM

Mike Whalen UNIVERSITY OF MINNESOTA, USA

Didar Zowghi UNIVERSITY OF TECHNOLOGY SYDNEY, AUSTRALIA

Program Committee

Dalal Alrajeh IMPERIAL COLLEGE LONDON, UK

Thomas Alspaugh GEORGETOWN UNIVERSITY, USA

Daniel Amyot UNIVERSITY OF OTTAWA, CANADA

João Araújo UNIVERSIDADE NOVA DE LISBOA, PORTUGAL

Nelly Bencomo ASTON UNIVERSITY, UK

Dan Berry UNIVERSITY OF WATERLOO, CANADA

Carl Chang IOWA STATE UNIVERSITY, USA

Daniela Damian UNIVERSITY OF VICTORIA, CANADA

Letícia Duboc UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO, BRAZIL

Steve Fickas UNIVERSITY OF OREGON, USA

Robert France COLORADO STATE UNIVERSITY, USA

Vincenzo Gervasi UNIVERSITÀ DI PISA, ITALY

Carlo Ghezzi POLITECNICO DI MILANO, ITALY

Martin Glinz UNIVERSITÄT ZÜRICH, SWITZERLAND

Olly Gotel INDEPENDENT RESEARCHER, USA

Sol Greenspan NATIONAL SCIENCE FOUNDATION, USA

Paul Grünbacher JOHANNES KEPLER UNIVERSITÄT LINZ, AUSTRIA

Robert Hall AT&T LABS RESEARCH, USA

Seok-Won Lee AJOU UNIVERSITY, REPUBLIC OF KOREA

Julio Leite PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO, BRAZIL

Soo Ling Lim UNIVERSITY COLLEGE LONDON AND BOURNEMOUTH UNIVERSITY, UK

Lin Liu TSINGHUA UNIVERSITY, CHINA

Walid Maalej UNIVERSITÄT HAMBURG, GERMANY

Patrick Mäder TECHNISCHE UNIVERSITÄT ILMENAU, GERMANY

Neil Maiden CITY UNIVERSITY LONDON, UK

Nancy Mead SOFTWARE ENGINEERING INSTITUTE, CMU, USA

Ana Moreira UNIVERSIDADE NOVA DE LISBOA, PORTUGAL

Gunter Mussbacher MCGILL UNIVERSITY, CANADA

Barbara Paech UNIVERSITÄT HEIDELBERG, GERMANY

Liliana Pasquale LERO, THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE, IRELAND

Óscar Pastor López UNIVERSITAT POLITÈCNICA DE VALÈNCIA, SPAIN

Klaus Pohl UNIVERSITÄT DUISBURG-ESSEN, GERMANY

Björn Regnell LUNDS UNIVERSITET, SWEDEN

Roshanak Roshandel SEATTLE UNIVERSITY, USA

Julia Rubin IBM RESEARCH - HAIFA, ISRAEL

Mehrdad Sabetzadeh UNIVERSITÉ DE LUXEMBOURG, LUXEMBOURG

Motoshi Saeki TOKYO INSTITUTE OF TECHNOLOGY, JAPAN

Pete Sawyer LANCASTER UNIVERSITY, UK

Kurt Schneider LEIBNIZ UNIVERSITÄT HANNOVER, GERMANY

Alistair Sutcliffe UNIVERSITY OF MANCHESTER, UK

Thein Than Tun THE OPEN UNIVERSITY, UK

Jon Whittle LANCASTER UNIVERSITY, UK

Roel Wieringa UNIVERSITEIT TWENTE, NETHERLANDS

Andrea Zisman THE OPEN UNIVERSITY, UK

Industry Committee

Ian Alexander SCENARIO PLUS, UK

Joy Beatty SEILEVEL INC., USA

Richard Berntsson Svensson CHALMERS UNIVERSITY OF TECHNOLOGY AND GÖTEBORGS UNIVERSITET, SWEDEN

David Callele EXPERIENCE FIRST DESIGN INC., CANADA

Alan M. Davis COLORADO STATE UNIVERSITY, USA

Carlos Henrique Duarte NATIONAL BANK FOR ECONOMIC AND SOCIAL DEVELOPMENT, BRAZIL

Christof Ebert VECTOR CONSULTANCY SERVICES, GERMANY

Markus Flückiger ZÜHLKE ENGINEERING AG, SWITZERLAND

Smita Ghaisas PRINCIPAL SCIENTIST AT TCS TATA, INDIA

Frank Houdé DAIMLER AG, GERMANY

Beatrice Hwong SIEMENS CORPORATE RESEARCH, USA

Sami Jantunen LAPPEENRANTA UNIVERSITY OF TECHNOLOGY, FINLAND

Pete Jones PHONAK, SWITZERLAND

Kim Lauenroth ADESSO AG, GERMANY

Alistair Mavin ROLLS-ROYCE, UK

Andrew Muyanja ARMS TECHNOLOGIES LTD., UGANDA

Juha Savolainen DANFOSS POWER ELECTRONICS, DENMARK

Erik Simmons INTEL, USA

Anja Wever SOFTWARE EDUCATION, AUSTRALIA

External Reviewers, Research Papers:

| | | |
|--------------------|-------------------|-----------------------|
| Okhaide Akhigbe | Emitza Guzman | Amir Molzam Sharifloo |
| Andre Almeida | Tom-Michael Hesse | Wuliang Sun |
| Nimanthi Atukorala | Paul Hübner | Giordano Tamburrelli |
| Phillipa Bennett | Michael Jackson | Bastian Tenbergen |
| Markus Borg | Timo Johann | Irina Todoran |
| Isabel Brito | Alessia Knauss | Nelufar Ulfat-Bunyadi |
| Curtis Busby-Earle | Luigi Logrippo | Kris Welsh |
| Everton Cavalcante | Hua Ming | Krzysztof Wnuk |
| Eya Ben Charrada | Thorsten Merten | Dustin Wüest |
| Philippe Collet | Andreas Metzger | Jingwei Yang |
| Marian Daun | Shiva Nejati | Gabriele Zorn-Pauli |
| Parisa Ghazi | Peter Newman | |
| Miguel Goulão | Katsunori Oyama | |

External Reviewers, Industry Papers:

| | |
|-------------------------|----------------|
| Luiz Paulo Alves Franca | Andrew Muyanja |
|-------------------------|----------------|

We would like to take a moment to thank all the gracious donors, sponsors and partners of RE'14.

Donors



Sponsors



Media Partners



Contents

Frontmatter

| | |
|-----------------------------------|-----|
| Message from the Chairs | iii |
| Committees | v |
| Sponsors | xi |

Keynotes

| | |
|---|---|
| Startups and Requirements (Keynote) | |
| Anthony I. Wasserman — <i>Carnegie Mellon University, USA</i> | 1 |
| Now More Than Ever: Privacy and Security Are Required (Keynote) | |
| Annie I. Antón — <i>Georgia Tech, USA</i> | 2 |

Research Track

Empirical Studies in Elicitation

| | |
|---|----|
| How Practitioners Approach Gameplay Requirements? An Exploration into the Context of Massive Multiplayer Online Role-Playing Games | |
| Maya Daneva — <i>University of Twente, Netherlands</i> | 3 |
| Therapist-Centered Requirements: A Multi-method Approach of Requirement Gathering to Support Rehabilitation Gaming | |
| Cynthia Putnam and Jinghui Cheng — <i>DePaul University, USA</i> | 13 |
| Towards a Situation Awareness Design to Improve Visually Impaired Orientation in Unfamiliar Buildings: Requirements Elicitation Study | |
| Abdulrhman Alkhanifer and Stephanie Ludi — <i>Rochester Institute of Technology, USA</i> | 23 |

Formal Modeling and Analysis

| | |
|---|----|
| Supporting Early Decision-Making in the Presence of Uncertainty | |
| Jennifer Horkoff, Rick Salay, Marsha Chechik, and Alessio Di Sandro — <i>University of Trento, Italy; University of Toronto, Canada</i> | 33 |
| Integrating Exception Handling in Goal Models | |
| Antoine Cailliau and Axel van Lamsweerde — <i>Université Catholique de Louvain, Belgium</i> | 43 |
| Protos: Foundations for Engineering Innovative Sociotechnical Systems | |
| Amit K. Chopra, Fabiano Dalpiaz, F. Başak Aydemir, Paolo Giorgini, John Mylopoulos, and Munindar P. Singh — <i>Lancaster University, UK; Utrecht University, Netherlands; University of Trento, Italy; North Carolina State University, USA</i> | 53 |

Legal and Regulatory Requirements

| | |
|--|----|
| Automated Detection and Resolution of Legal Cross References: Approach and a Study of Luxembourg's Legislation | |
| Morayo Adedjouma, Mehrdad Sabetzadeh, and Lionel C. Briand — <i>University of Luxembourg, Luxembourg</i> | 63 |

| | |
|--|-----|
| Goal-Oriented Compliance with Multiple Regulations | |
| Sepideh Ghanavati, André Rifaut, Eric Dubois, and Daniel Amyot — <i>CRP Henri Tudor, Luxembourg; University of Ottawa, Canada</i> | 73 |
| Identifying and Classifying Ambiguity for Regulatory Requirements | |
| Aaron K. Massey, Richard L. Rutledge, Annie I. Antón, and Peter P. Swire — <i>Georgia Tech, USA</i> | 83 |
| Handling Change and Evolution | |
| An Approach for Decision Support on the Uncertainty in Feature Model Evolution | |
| Le Minh Sang Tran and Fabio Massacci — <i>University of Trento, Italy</i> | 93 |
| Maintaining Requirements for Long-Living Software Systems by Incorporating Security Knowledge | |
| Stefan Gärtner, Thomas Ruhroth, Jens Bürger, Kurt Schneider, and Jan Jürjens — <i>Leibniz Universität Hannover, Germany; TU Dortmund, Germany</i> | 103 |
| Rationalism with a Dose of Empiricism: Case-Based Reasoning for Requirements-Driven Self-Adaptation | |
| Wenyi Qian, Xin Peng, Bihuan Chen, John Mylopoulos, Huanhuan Wang, and Wenyun Zhao — <i>Fudan University, China; University of Trento, Italy</i> | 113 |
| Traceability | |
| TiQi: Towards Natural Language Trace Queries | |
| Piotr Pruski, Sugandha Lohar, Rundale Aquanette, Greg Ott, Sorawit Amornborvornwong, Alexander Rasin, and Jane Cleland-Huang — <i>DePaul University, USA</i> | 123 |
| Traceability-Enabled Refactoring for Managing Just-In-Time Requirements | |
| Nan Niu, Tanmay Bhowmik, Hui Liu, and Zhendong Niu — <i>University of Cincinnati, USA; Mississippi State University, USA; Beijing Institute of Technology, China</i> | 133 |
| Supporting Traceability through Affinity Mining | |
| Vincenzo Gervasi and Didar Zowghi — <i>University of Pisa, Italy; University of Technology Sydney, Australia</i> | 143 |
| Discovering Requirements | |
| How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews | |
| Emitzia Guzman and Walid Maalej — <i>TU München, Germany; University of Hamburg, Germany</i> | 153 |
| Scaling Requirements Extraction to the Crowd: Experiments with Privacy Policies | |
| Travis D. Breaux and Florian Schaub — <i>Carnegie Mellon University, USA</i> | 163 |
| Discovering Affect-Laden Requirements to Achieve System Acceptance | |
| Alistair Sutcliffe, Paul Rayson, Christopher N. Bull, and Pete Sawyer — <i>Lancaster University, UK</i> | 173 |
| Security and Privacy Requirements | |
| Hidden in Plain Sight: Automatically Identifying Security Requirements from Natural Language Artifacts | |
| Maria Riaz, Jason King, John Slankas, and Laurie Williams — <i>North Carolina State University, USA</i> | 183 |
| Managing Security Requirements Patterns using Feature Diagram Hierarchies | |
| Rocky Slavin, Jean-Michel Lehker, Jianwei Niu, and Travis D. Breaux — <i>University of Texas at San Antonio, USA; Carnegie Mellon University, USA</i> | 193 |
| Engineering Topology Aware Adaptive Security: Preventing Requirements Violations at Runtime | |
| Christos Tsigkanos, Liliana Pasquale, Claudio Menghi, Carlo Ghezzi, and Bashar Nuseibeh — <i>Politecnico di Milano, Italy; Lero, Ireland; Open University, UK</i> | 203 |
| Communicating Requirements | |
| Openness and Requirements: Opportunities and Tradeoffs in Software Ecosystems | |
| Eric Knauss, Daniela Damian, Alessia Knauss, and Arber Borici — <i>Chalmers, Sweden; University of Gothenburg, Sweden; University of Victoria, Canada</i> | 213 |
| RISDM: A Requirements Inspection Systems Design Methodology: Perspective-Based Design of the Pragmatic Quality Model and Question Set to SRS | |
| Shinobu Saito, Mutsuki Takeuchi, Setsuo Yamada, and Mikio Aoyama — <i>NTT DATA, Japan; NTT, Japan; Nanzan University, Japan</i> | 223 |
| Tackling the Requirements Jigsaw Puzzle | |
| Maria Pinto-Albuquerque and Awais Rashid — <i>Lisbon University Institute, Portugal; Lancaster University, UK</i> | 233 |

Automated Support for Eliciting Requirements

Automated Support for Combinational Creativity in Requirements Engineering

- Tanmay Bhowmik, Nan Niu, Anas Mahmoud, and Juha Savolainen — *Mississippi State University, USA; University of Cincinnati, USA; Danfoss, Denmark* 243

Automated Extraction and Visualization of Quality Concerns from Requirements Specifications

- Mona Rahimi, Mehdi Mirakhorli, and Jane Cleland-Huang — *DePaul University, USA* 253

Requirements Management Concerns

Language Extended Lexicon Points: Estimating the Size of an Application using Its Language

- Leandro Antonelli, Gustavo Rossi, Julio Cesar Sampaio do Prado Leite, and Alejandro Oliveros — *Universidad Nacional de La Plata, Argentina; PUC-Rio, Brazil; Universidad Argentina de la Empresa, Argentina* 263

The Role of Legal Expertise in Interpretation of Legal Requirements and Definitions

- David G. Gordon and Travis D. Breaux — *Carnegie Mellon University, USA* 273

Evaluating the Business Value of Information Technology: Case Study on Game Management System

- Harri Töhönen, Marjo Kauppinen, and Tomi Männistö — *Aalto University, Finland; University of Helsinki, Finland* 283

Quality Goals

Non-functional Requirements as Qualities, with a Spice of Ontology

- Feng-Lin Li, Jennifer Horkoff, John Mylopoulos, Alexander Borgida, Renata S. S. Guizzardi, Giancarlo Guizzardi, and Lin Liu — *University of Trento, Italy; Rutgers University, USA; Federal University of Espírito Santo, Brazil; Tsinghua University, China* 293

Quality Requirements Elicitation Based on Inquiry of Quality-Impact Relationships

- Farnaz Fotrousi, Samuel A. Fricker, and Markus Fiedler — *Blekinge Institute of Technology, Sweden* 303

Tool Demonstrations and Posters

Nòmos 3: Reasoning about Regulatory Compliance of Requirements

- Silvia Ingolfo, Alberto Siena, and John Mylopoulos — *University of Trento, Italy* 313

GUITAR: An Ontology-based Automated Requirements Analysis Tool

- Tuong Huan Nguyen, John Grundy, and Mohamed Almorsy — *Swinburne University of Technology, Australia* 315

Simulation-Based Requirements Discovery for Smart Driver Assistive Technologies

- Andreas Gregoriades, Maria Pampaka, and Alistair Sutcliffe — *European University Cyprus, Cyprus; University of Manchester, UK* 317

EAM: Ecosystemability Assessment Method

- Eric Knauss and Imed Hammouda — *Chalmers, Sweden; University of Gothenburg, Sweden* 319

Combined Goal and Feature Model Reasoning with the User Requirements Notation and jUCMNav

- Yanji Liu, Yukun Su, Xinshang Yin, and Gunter Mussbacher — *McGill University, Canada* 321

Decisively: Application of Quantitative Analysis and Decision Science in Agile Requirements Engineering

- Sanjaya Kumar Saxena and Rachna Chakraborty — *GrayPE Systems, India* 323

VARED: Verification and Analysis of Requirements and Early Designs

- Julia Badger, David Throop, and Charles Claunch — *NASA, USA; Boeing, USA* 325

Structured Multi-view Modeling by Tabular Notation

- Xiuna Zhu, Dongyue Mou, and Daniel Ratiu — *TU München, Germany; fortiss, Germany* 327

Efficient Visual Notations for Efficient Stakeholder Communication

- Ralf Laue, Frank Hogrebe, Boris Böttcher, and Markus Nüttgens — *University of Applied Sciences Zwickau, Germany; Hessische Hochschule für Polizei und Verwaltung Wiesbaden, Germany; University of Hamburg, Germany* 329

Symbolic Verification of Requirements in VRS System

- Oleksandr Letychevskyi and Thomas Weigert — *Glushkov Institute of Cybernetics, Ukraine; Uniquesoft LLC, USA* 331

Business Application Modeler: A Process Model Validation and Verification Tool

- Sören Witt, Sven Feja, Andreas Speck, and Christian Hadler — *Kiel University, Germany* 333

Industry Track

Lightweight RE Methods

| | |
|--|-----|
| Handling Design-Level Requirements across Distributed Teams: Developing a New Feature for 12 Danish Mobile Banking Apps Lars Bruun, Mikkel Bovbjerg Hansen, Jørgen Bøndergaard Iversen, Jens Bæk Jørgensen, and Bjarne Knudsen — <i>Bankdata, Denmark; Mjølner Informatics, Denmark</i> | 335 |
| Experience of Pragmatically Combining RE Methods for Performance Requirements in Industry Rebekka Wohlrab, Thijmen de Gooijer, Anne Koziolek, and Steffen Becker — <i>ABB Research, Sweden; KIT, Germany; University of Paderborn, Germany</i> | 344 |
| Lightweight Requirements Engineering Assessments in Software Projects Daniel Rapp, Anne Hess, Norbert Seyff, Peter Spörri, Emmerich Fuchs, and Martin Glinz — <i>Zühlke Management Consultants, Switzerland; Fraunhofer IESE, Germany; University of Zurich, Switzerland</i> | 354 |

Stakeholder Collaboration

| | |
|---|-----|
| Capturing and Sharing Domain Knowledge with Business Rules: Lessons Learned from a Global Software Vendor Walid Maalej and Smita Ghaisas — <i>University of Hamburg, Germany; Tata Consultancy Services, India</i> | 364 |
| Building a National E-Service using Sentire: Experience Report on the Use of Sentire: A Volere-Based Requirements Framework Driven by Calibrated Personas and Simulated User Feedback Chris Porter, Emmanuel Letier, and M. Angela Sasse — <i>University College London, UK</i> | 374 |
| Competition and Collaboration in Requirements Engineering: A Case Study of an Emerging Software Ecosystem George Valençá, Carina Alves, Virgínia Heimann, Slinger Jansen, and Sjaak Brinkkemper — <i>Federal Rural University of Pernambuco, Brazil; Federal University of Pernambuco, Brazil; Utrecht University, Netherlands</i> | 384 |
| The Effect of Variability Modeling on Requirements Satisfaction for the Configuration and Implementation of Off-The-Shelf Software Packages Amanda Rubython and Neil Maiden — <i>City&Guilds Kineo, UK; City University London, UK</i> | 394 |

RE in Practice: Experiences from the Field I

| | |
|---|-----|
| Modelling Sustainability in a Procurement System: An Experience Report Camilla Bomfim, Wesley Nunes, Letícia Duboc, and Marcelo Schots — <i>State University of Rio de Janeiro, Brazil; Federal University of Rio de Janeiro, Brazil</i> | 402 |
| A Case Study using a Protocol to Derive Safety Functional Requirements from Fault Tree Analysis Luiz Eduardo Galvão Martins and Tiago de Oliveira — <i>Federal University of São Paulo, Brazil</i> | 412 |
| The DODT Tool Applied to Sub-Sea Software Tor Stålhane and Tormod Wien — <i>Norwegian University of Science and Technology, Norway; ABB Research, Norway</i> | 420 |

RE in Practice: Experiences from the Field II

| | |
|--|-----|
| Towards Feature-Oriented Requirements Validation for Automotive Systems Jiale Zhou, Yue Lu, Kristina Lundqvist, Henrik Lönn, Daniel Karlsson, and Bo Liwång — <i>Mälardalen University, Sweden; Volvo, Sweden; Swedish Radiation Safety Authority, Sweden</i> | 428 |
| Product Knowledge Configurator for Requirements Gap Analysis and Customizations Preethu Rose Anish and Smita Ghaisas — <i>Tata Consultancy Services, India</i> | 437 |
| Reassessing the Pattern-Based Approach for Formalizing Requirements in the Automotive Domain Predrag Filipovikj, Mattias Nyberg, and Guillermo Rodriguez-Navas — <i>Mälardalen University, Sweden; Scania, Sweden</i> | 444 |

Doctoral Symposium

| | |
|--|-----|
| From Architecture to Requirements: Relating Requirements and Architecture for Better Requirements Engineering Feng Chen — <i>University of Limerick, Ireland; Lero, Ireland</i> | 451 |
|--|-----|

| | |
|--|-----|
| Quantification of Social Sustainability in Software Maryam Al Hinai — <i>University of Leicester, UK</i> | 456 |
| Improving Collaborative and Post-WIMP Systems through Requirements Specification Miguel A. Teruel — <i>University of Castile-La Mancha, Spain</i> | 461 |
| Stakeholders' Social Interaction in Requirements Engineering of Open Source Software Tanmay Bhowmik — <i>Mississippi State University, USA</i> | 467 |
| Aligning Services and Requirements with User Feedback Muneera Bano — <i>University of Technology Sydney, Australia</i> | 473 |
| Requirements Development and Management of Embedded Real-Time Systems Jiale Zhou — <i>Mälardalen University, Sweden</i> | 479 |
| Context-Sensitive Information Security Risk Identification and Evaluation Techniques Dan Ionita — <i>University of Twente, Netherlands</i> | 485 |
| Business Processes and Regulations Compliance Management Technology Ilze Buksa — <i>Riga Technical University, Latvia</i> | 489 |
| Creative Strategic Scenarios for Preparation to Requirements Evolution Marília Guterres Ferreira — <i>PUC-Rio, Brazil</i> | 494 |

Panel

| | |
|--|-----|
| Ready-Set-Transfer! Technology Transfer in the Requirements Engineering Domain (Panel) Jane Huffman Hayes and Didar Zowghi — <i>University of Kentucky, USA; University of Technology Sydney, Australia</i> | 500 |
|--|-----|

Author Index

Startups and Requirements (Keynote)

Anthony I. Wasserman
Carnegie Mellon University, USA

ABSTRACT

Tech startups typically approach requirements gathering differently from the process that is used to develop requirements in existing businesses and for existing products. Many of these startups operate in "stealth mode", taking care to minimize the number of people with whom they share their innovative ideas. It's common for these startups to create a succession of early releases, and apply user feedback from those releases to evolve the features and functions of their products.

This talk describes various approaches used by tech startups to define product requirements, and contrasts those approaches with those used in more traditional requirements engineering activities.

BIOGRAPHY

Anthony I. (Tony) Wasserman is a Professor of Software Management Practice at Carnegie Mellon University - Silicon Valley, and the Executive Director of its Center for Open Source Investigation (COSI). In Spring, 2014, he was Shaw Visiting Professor in the Department of Computer Science at the National University of Singapore.

Tony is best known as the Founder and CEO of Interactive Development Environments (IDE), which built the innovative Software through Pictures (S^tP) multiuser modeling environment. IDE was one of the first companies to include open source software in a commercial software product. In 2000, Tony became VP of Bluestone Software, where he led the creation of the award-winning open source Total-e-Mobile

toolkit for building mobile web apps. Prior to that, he was a Professor at the University of California, San Francisco.

Tony is currently a Director of the Open Source Initiative, and a member of the Board of Advisors of Open Source for America. He is a member of the Executive Advisory Board of CollabNet, Inc.

He earned a Ph.D. in Computer Sciences from the University of Wisconsin - Madison and a B.A. in mathematics and physics from the University of California, Berkeley. He is a Life Fellow of the IEEE and a Fellow of the ACM. He was the 2012 recipient of the Distinguished Educator Award from the IEEE Technical Council on Software Engineering, and the 2013 recipient of the Influential Educator Award from the ACM Special Interest Committee on Software Engineering (SIGSOFT).



Now More Than Ever: Privacy and Security Are Required (Keynote)

Annie I. Antón
Georgia Institute of Technology, USA

ABSTRACT

Properly protecting information is in all our best interests, but it is a complex undertaking. The fact that regulation is often written by non-technologists, introduces additional challenges and obstacles. Moreover, those who design systems that collect, store, and maintain sensitive information have an obligation to design systems holistically within this broader context of regulatory and legal compliance. There are questions that should be asked when developing new requirements for information systems. For example, how do we build systems to handle data that must be kept secure and private when relevant regulations tie your hands? When building a system that maintains health or financial records for a large number of people, what do we need to do to protect the information against theft and abuse, keep the information private, AND at the same time, satisfy all governing privacy/security laws and restrictions? Moreover, how do we know that we've satisfied those laws? How do we monitor for compliance while ensuring that we're monitoring the right things? And, how do you accomplish all this in a way that can be expressed clearly to end-users and legislators (or auditors) so they can be confident you are doing the right things? We've been working on technologies to make these tasks simpler, and in some senses, automatic. In this talk, I will describe some of the research that we have been conducting to address these problems.

BIOGRAPHY

Dr. Annie I. Antón is a Professor in and Chair of the School of Interactive Computing at the Georgia Institute of Technology in Atlanta. She has served the national defense and intelligence communities in a number of roles since being selected for the IDA/DARPA Defense Science Study Group in 2005–2006. Her current research focuses on the specification of complete, correct behavior of software systems

that must comply with federal privacy and security regulations. She is founder and director of ThePrivacyPlace.org. Antón currently serves on various boards, including: the U.S. DHS Data Privacy and Integrity Advisory Committee, an Intel Corporation Advisory Board, and the Future of Privacy Forum Advisory Board. She is a former member of the CRA Board of Directors, the NSF Computer & Information Science & Engineering Directorate Advisory Council, the Distinguished External Advisory Board for the TRUST Research Center at U.C. Berkeley, the DARPA ISAT Study Group, the USACM Public Council, the Advisory Board for the Electronic Privacy Information Center in Washington, DC, the Georgia Tech Alumni Association Board of Trustees, the Microsoft Research University Relations Faculty Advisory Board, the CRA-W, and the Georgia Tech Advisory Board (GTAB). Prior to joining the faculty at Georgia Tech, she was a Professor of Computer Science in the College of Engineering at the North Carolina State University. Antón is a three-time graduate of the College of Computing at the Georgia Institute of Technology, receiving a Ph.D. in 1997 with a minor in Management & Public Policy, an M.S. in 1992, and a B.S. in 1990 with a minor in Technical and Business Communication.



How Practitioners Approach Gameplay Requirements? An Exploration into the Context of Massive Multiplayer Online Role-Playing Games

Maya Daneva

Dept. of Computer Science
University of Twente
Enschede, The Netherlands
m.daneva@utwente.nl

Abstract—Gameplay requirements are central to game development. In the business context of massive multiplayer online role-playing games (MMOGs) where game companies' revenues rely on players' monthly subscriptions, gameplay is also recognized as the key to player retention. However, information on what gameplay requirements are and how practitioners 'engineer' them in real life is scarce. This exploratory study investigates how practitioners developing MMOGs reason about gameplay requirements and handle them in their projects. 12 practitioners from three leading MMOGs-producing companies were interviewed and their gameplay requirements documents were reviewed. The study's most important findings are that in MMOG projects: (1) gameplay requirements are co-created with players, (2) are perceived and treated by practitioners as sets of choices and consequences, (3) gameplay is endless within a MMOG, and while gameplay requirements do not support any game-end goal, they do support a level-end goal, (4) 'paper-prototyping' and play-testing are pivotal to gameplay validation, (5) balancing the elements of the gameplay is an on-going task, perceived as the most difficult and labor-consuming, (6) gameplay happens both in-game and out-of-the game. We conclude with discussion on validity threats to our results and on implications for research and practice.

Index Terms—Massive Multiplayer Online Role-Playing Games, Gameplay design, Play-centric requirements engineering, Play-testing, Gameplay balancing, Exploratory case study, Empirical research method.

I. INTRODUCTION

A massive multiplayer online role-playing game (MMOG), such as World of Warcraft or EverQuest, is an entertainment software system that provides an online environment to a large number of geographically distributed players to simultaneously participate in game communities and engage in social interaction and game activities, e.g. going on missions, fighting dangerous creatures (monsters), attacking castles, trading goods. Game Market Research firms (e.g. DFC Intelligence and Newzoo) indicate MMOGs as the fastest growing industry, generating worldwide revenue expected to reach 29 billion US\$ by 2016 [1,2]. Playing MMOGs is also the most intense collective human activities on the planet, including hundreds of millions of players. As the 21st century's MMOG online worlds get vast and complex, designing them usually is long and expensive [1,2]. Central to the MMOG design is the so-

called 'gameplay' which is most generally regarded in MMOG literature (e.g. [11]) as to what players and non-player characters (NPCs) in the game do that is entertaining (also known as 'fun'). For example, the gameplay in the World of Warcraft series (Level 90+), a popular MMOG game published by Blizzard Entertainment, is collaboratively planning raids and defeating monsters. Currently, gameplay as part of MMOG design processes have been empirically studied primarily in the fields of game design (e.g. [3,4,5,6,7]), human-computer interaction (e.g. [8,9,10]), social behavior (e.g. [11]), media communication (e.g. [33]), psychology (e.g. [12]) and education (e.g. [34]). In the field of requirements engineering (RE), games only relatively recently became a topic of active research (e.g. [13,14,36,37,38]) and whatever empirical evidence got published covered design of video games [13,36], console-based [37,38] and mobile phone games [14,38]. While the published research on RE for game systems [13,14,15] had acknowledged the paramount importance of gameplay requirements, it also found a knowledge gap in our understanding of these requirements, and in turn called for further research in the area. This paper directly responds to this call and extends the conversation on gameplay requirements to the realm of MMOGs. To the best of our knowledge no study has been published on the particular context of MMOGs.

We set out to execute an exploratory study in order to investigate how practitioners developing MMOGs reason about gameplay requirements and handle them in their projects. Practitioners from three leading companies in MMOGs development and publishing have been interviewed and their gameplay requirements documents were reviewed in one-on-one walkthroughs with the researcher.

The key result of the study is a contextualized description of the practitioners views on gameplay requirements and on the ways they are coped with. The findings are compared with those from published relevant studies in the fields of human-computer interaction, social behavior, and psychology, and some implications are drawn for research and practice.

In the next sections we provide background on MMOGs and related work on play-centric game development and RE. We then describe our research process, its execution, and its results. This is followed by our discussion on implications for research and practice and our evaluation of validity threats.

II. BACKGROUND AND RELATED WORK

A. MMOG applications

This section informs less familiar readers on MMOG as online businesses and as systems. MMOG sites are powerhouses of e-commerce, forming part of the entertainment services industry. They usually use two business models, subscription-based and ‘freemium’. The first generates profit out of a growing long-term subscriber base and, hence, player retention is key to MMOG producing companies’ financial health. In this model, subscribers buy the game (client) software for, e.g. 50 Euro, and pay a monthly fee of e.g. 15 Euro, to access the game online (server). In contrast, the freemium model assumes the presence of a huge base of free clients and a percentage of it to turn into paid clients at some point of the process of playing the game. Usually, in this model the producer gives away the game’s core functionality for free, while offering upgrades to add certain features for a small fee. Converting free players to paid ones at a consistent rate is strategic to such a producer. In both business models, to trill players and keep them playing, MMOG companies vigorously embrace innovation in both signal processing and game design, and commit to deliver new, high quality game contents (e.g. extensions, patches) on regular basis.

The MMOG functionality provides players with an extensive range of game abilities, including character design options, weapons, tools, statistics, all of which grow in number and intensity proportionate with player’s progress through the game. In a MMOG, players live through their characters (avatars), and compete with other players’ avatars and non-player characters (NPCs), e.g. dwarfs. For players to start the game, they must create their avatars first, by choosing a name, gender, race, social class, and facial features. Avatars necessarily join one of two conflicting factions – good and evil, that build up the storyline for each player. Choosing a faction implies for players to enter the game through this faction’s own home city, and that they design their avatars based on the options available to the faction exclusively. As a player’s avatar advances throughout the game, its abilities gradually grow as well as the number of potential approaches to in-game situations, e.g. quests, fighting, crafting and teaming-up. While playing, players can switch between two perspectives: the first person and the third person perspective. Also, based on whether players fight against other players or NPCs, fighting happens respectively on Player versus Environment or Player versus Player servers. In both, players use weapons and spells to deal damage to the target and upon success they are awarded experience points that are critical to moving from one game level to another. Preparing for fights and fighting gets increasingly more social and group-based, as levels advance. Once a player passes the end level of a MMOG, the only way to continue playing endlessly is by joining a group.

B. Play-centric Game Design, Requirements Engineering, and Gameplay

MMOG companies usually use play-centric approaches to game development that span from idea conceptualization to

completion. Central to them is the goal to achieve playable and satisfying game experience; this means continually keeping the player experience as the driver in making design choices through the game development process, and testing the gameplay that these choices create with target players at each development stage [3]. In a play-centric development context, RE is an agile process as it’s not limited to the early stages of game conceptualization; instead, it accompanies the play-testing and the prototyping activities as assumptions about player requirements might render unrealistic or new requirements may be discovered; Also, RE is an on-going process because it goes well beyond MMOG launch and active play as companies keep collecting players’ feedback via online communities, which is an important way to get requirements for upgrades and new releases.

Prior research on RE for games did acknowledge the play-centric nature of requirements processes (e.g. [4,5]) and the prominent role that enjoyability, fun, emotions and gameplay take in game development [5,6,8,38]. However, in the field of RE, very little research focused on gameplay specifically. The 2010 systematic review on software engineering for computer games [19] found 37 empirical studies on RE for games. However, while reviewing them as part of preparing this paper, we found that only two [13,14] included gameplay requirements as a sub-topic in their analyses, in the context of video games and mobile games development, respectively, e.g. the study of Callele et al [13] presents some techniques practitioners used for gameplay prototyping in video game design. Both the studies of Alves et al [14] and Callele et al [13] point to these requirements as a particularly challenging area for which more research is needed. However, none of these studies elaborates on what gameplay requirements are in detail, nor dealt with gameplay requirements exclusively nor considered gameplay in MMOG context. We also searched the Scopus digital library for relevant RE studies published after 2010. We found three empirical studies [36,37,38] on RE for games: [36] is concerned with console-based games (e.g. Wii), [37] – with serious games, and [38] – with the role of RE in (non-MMOG) game development including various platforms. Among these studies, [38] is the only one in which gameplay requirements and related concepts such as game mechanics, prototyping of requirements and balancing games’ rules are discussed explicitly.

In the fields of human-computer interaction and social psychology, researchers investigated usability [20], playability [21], game engagement [11,12], gameplay experience [29] and aesthetics aspects [10] of video games and MMOGs, with a particular focus on gameplay. In the field of game design, gameplay was conceptualized by employing layered models, e.g. LeBlanc’s Mechanistic-Dynamics-Aesthetics [4], or gameplay structures that mean the players’ interactions in the multi-play happening in a game [6]. Approaches also have been developed to capture and document gameplay design patterns, e.g. the game ontology project [5] that employs a hierarchical structure to describe gameplay elements from players’ perspective, and the 400 rules project [7] that stores and shares knowledge in ‘rules’ format, from professional

game designers. These studies on gameplay can be useful for understanding games and games mechanics but they do not directly address gameplay requirements and how they are handled. However, following Callele et al [13], we think that even if these studies do not address RE, the techniques they proposed could possibly form an interesting extension to the RE approaches already available to practitioners.

III. THE EXPLORATORY STUDY RESEARCH PLAN

A. Research Goal and Research Questions

To understand how requirements elicitation and discovery happens in MMOG projects, we set out to answer the following research questions (RQ):

RQ1: What gameplay requirements mean to practitioners involved in MMOG development projects?

RQ2: What MMOG project artefacts are concerned with gameplay requirements? and

RQ3: How do practitioners manage the process of gameplay RE?

Our research design followed Yin's guidelines for exploratory studies [22]. As Bensabat et al [24] suggest, a case study is a particularly suitable research method to situations in which (1) researchers study socially constructed processes in systems development projects and seek to achieve as good as possible grasp of reality, and (2) researchers want to answer 'how' and 'why' questions and comprehend the nature and the complexity of the processes of interest. Drawing upon this, we expected the case study research method would let us earn a rich and contextualized understanding of the practitioners' meanings behind the concepts of 'gameplay' and 'gameplay requirements' and the ways in which practitioners approach the engineering tasks essential to these requirements.

B. The Empirical Research Process

We designed a research process including semi-structured open-end in-depth interviews with 12 practitioners from three producers that are highly visible in the MMOG market. Because of confidentiality agreements, we would not be able to provide any information on the country location of these producers, nor on their organizations' size. However, the MMOG companies shared the following common characteristics: (i) they all use the subscription-based business model and player retention is their key strategic objective; (ii) all three games included in the study are known to be profitable ventures; (iii) each company had at least one million subscribers, (iv) they all follow play-centric approaches [3] to MMOG design; (v) all three games have a common goal, namely to increase the fitness of the player's character, which is endless in nature and not bound to a particular game level (because keeping players playing is in essence player retention); (vi) the companies all built in-game and out-of-the-game communities of players and have the continuous improvement of the players' social collaboration as an underlying motivation for MMOG development projects; they produce packs, extensions, enhancements, on annual basis.

Table 1 summarizes the information about the participating companies and practitioners. Their job titles and project roles

were diverse, including various artistic specialists, level designer, play-testing manager, community development manager, playability analyst, gameplay analyst, senior software engineers (e.g. on client and server sides).

TABLE 1. THE COMPANIES AND THE INTERVIEW PARTICIPANTS.

| Company name | Details | Interviewee designation | Estimated work time spent on gameplay requirements tasks |
|--------------|---|---|--|
| TechGame | North-America-based game developer and publisher with a Player Support Centre in Europe and an international player base | 1 Level Designer 1 Playability Analyst 1 Senior Software Engineer – Server Systems 1 Senior Software Engineer – Client Systems | 75% 30% 30% 50% |
| ArtGame | North-America-based game developer and publisher with a Player Support Centres in Europe and Asia, and an international player base | 1 Environment Texture Artist 1 Character Artist 1 Project Lead 1 Play-testing Manager | 50% 50% 50% 100% |
| YouGame | East-Asia-based game developer and publisher with operations in Europe and an international player base | 1 Community Dev. Manager 1 Visual Effect Artist 1 Project Lead 1 Gameplay Analyst | 100% 50% 50% 100% |

Our study was carried out in four steps: (1) Prepare an interview guide following the guidelines in [23]; (2) Do a pilot interview with one practitioner to check the applicability of the guide to real-life context; (3) Carry out interviews with practitioners according to the finalized questionnaire; (4) Follow-up with those participants that possess deeper knowledge or a specific perspective. We note that after the first interview (the pilot), there were no substantial changes in the interview guide and for this reason, we included the pilot interview as a legitimate one in our data analysis (as suggested by [14]). The interview guide is receivable from the author upon requests of interested readers.

The time each interview took was between 90 and 100 minutes. Four interviews were face-to-face, and eight - over a conference bridge with shared presentation screen facility used for walking through a gameplay document that the respective interviewee presented to the researcher. As strict confidentiality agreements were discussed before the interviewees gave their consent to participate, each interviewee was informed by the author on the research process and on how data will be handled. The author provided each participant with the list of interview questions two weeks before the interview date. Each interviewee was also asked to bring a gameplay document to be used for illustration purposes during the interview, so that the researcher makes sure she understands what the practitioner means when using a specific gameplay-requirements-related concept. Each interviewee came prepared to the interview, which was key to the effective use of the interview time slots.

1) *Selecting and Characterizing the Participants:* We used two selection criteria for practitioners' participation. The participants should: (1) have exposure to the development context of MMOGs; and (2) be involved in RE tasks as part of their professional responsibility in the company. 'Involved' meant that at least 30% of the work time that a practitioner logs in a project goes to gameplay requirements. As Table 1 shows, we had two practitioners (out of 12) that spent 30% of their work time on gameplay requirements tasks, three that said gameplay requirements were the very core of their jobs and they spent 100% of their time on them, and seven who spent 50%-75% of their work time on gameplay requirements. Tasks could broadly range from writing gameplay requirements to attending review meetings, building prototypes and providing feedback. The practitioners were chosen because their jobs pertained to this study's research context. Their experience included at least three projects concerning the MMOG produced by their companies. All practitioners were avid players of their respective company's MMOG. Their playing experience was at least 6 years. We note that the hiring policies in all three companies explicitly stated preferences for job candidates who had playing experience. In fact, all 12 practitioners had met this condition when they applied for their positions in their respective companies. Last but not least, the practitioners were excited about this exploratory study, they felt enthusiastic about winning a researcher's interest in their professional activity. They all thought, researchers could help industry by promoting careers in game design, to the students they teach.

To get access to the practitioners, the author used as mediator a member of the International Game Development Association (IGDA, www.igda.org). The mediator chose to remain anonymous and not participate as a co-author of this paper. He was a former colleague of the author and was instrumental to hook her up with relevant parties in the three companies. Thanks to the mediator, the author also made a visit at the European branch of one of the companies and executed the face-to-face interviews. The mediator suggested a broad range of possible contacts however the author used purposive sampling [23] to select the 12 interviewees so that they represent a broad variety of viewpoints, roles and contributions to gameplay RE for MMOG.

2) *Our data collection:* We used the method of 'focused semi-structured' interview [23] for data collection. 'Focused' means that (1) gameplay, (2) gameplay requirements and (3) managing gameplay RE were the focus of all conversations with the 12 practitioners. 'Semi-structured' means that a list of interview questions is prepared in advance, however, during the interview the researcher has the right to ask new questions (e.g. to explore a specific perspective on an issue and probe for details) or to drop existing questions based on his/her judgment of the opportunities to extract rich contextual information in the conversation with the interviewee. Choosing the semi-structured interview method ensured our data collection was shaped by both the participant's own understandings of gameplay and gameplay requirements as

well as the researcher's interests. Also, it allowed us to take unexpected routes as opportunity arose. We note that interview-based exploratory studies usually are intended to promote self-disclosure and that is what we were after in this work. As in [23], interview studies are not used to provide statistically generalizable results applicable to all professionals with profiles similar to those of the practitioners in a specific study. The intention of the exploratory case study is not to infer, but to understand and not to generalize, but to determine a possible range of views. Therefore, in this study we will adopt, based on the recommendations in [23], the criterion of transferability as a useful measure of validity. Transferability asks for whether the results are presented in a way that allows other researchers and practitioners to evaluate if the findings apply to their contexts.

3) *Our data analysis:* Our data analysis was inspired by the guidelines in Charmaz' grounded theory (GT) method [25], a qualitative approach commonly deployed in empirical studies in social sciences to construct general propositions (called a "theory" in this approach) from verbal data. The GT analysis is exploratory in nature and suitable to research settings where the researcher does not want to use pre-conceived ideas, and instead is driven by the desire to capture all facets of the collected data and to allow the propositions to emerge from the data. The essence of the GT guidelines is in making analytic sense of the interview data by means of coding and constant comparison of pieces of data that were collected in the case study. Constant comparison means that the data from an interview is constantly compared to the data already collected from previously held interviews. The data analysis process deployed in this study, leveraged the author's prior experience in empirical studies using GT (e.g. [26]). As in these previous studies, our data analysis in this study started with reading interview texts and labeling a portion of the text – a phrase or a paragraph, with a 'coding' word. The 'codes' mark the meaning of the respective portion of the interview text to a specific research question. A code can be a concept (e.g. 'look and feel', 'playability', 'player experience') or an activity (e.g. 'balancing the gameplay', 'play-testing', 'paper-prototyping'). All pieces of text that relate to the same code are then clustered in order to analyze it in a consistent and systematic way. The results of the data analysis and the discussion on them are in Sections IV and V.

IV. RESULTS

In this section, the codes - concepts and activities, are marked by using an underlined italicized font in their first appearance, e.g. play-testing is marked as *play-testing*. As it is usual in qualitative studies, we supplement the findings with interviewees' quotations. These are given in italic.

A. *The Meanings of Gameplay and Gameplay Requirements*

From RE perspective, our analysis suggests gameplay requirements refers to a group of requirements - and their interactions, that when satisfied together, deliver player's experience that matches or exceeds the players' expectation.

In our participants' views, the interactions between the requirements in the group heavily contribute to the overall effect on player's experience. While our participants agreed on this understanding of gameplay requirements, they differed regarding how they think of the term 'gameplay' itself. Our findings suggest no consensus on what gameplay is. The interviewees hinted to three ways to conceptualize gameplay: as a process, as an artefact, and as a relationship between game creators and players.

Five practitioners thought of gameplay as a process that a player has to go through in order to achieve the end of a level. In essence, they made a clear distinction between two types of gameplay-as-a-process: (1) 'within level' which is bounded to a level in the game (e.g. level 2 in a game of 99 levels); and (2) 'end level' gameplay, which is the one that happens once players exhaust all levels and start consuming the so-called 'end-level game content' (e.g. once one is at level 99). At each level, the gameplay process is perceived as a series of tasks arranged in three groups: preparing the avatar for coping with the in-level challenges, execution, and wrap-up. The tasks may vary broadly in terms of contents, complexity, and player's time and skill demands. However, once the end level is achieved, the gameplay process' goal is endless and focused solely on equipping the avatar as well as possible. E.g. the endgame experience is focused on organizing for large-scale heavily-coordinated attacks among up to 60 players, where players are expected to commit significant amount of time (up to 18 hours weekly) in configuring the avatar in a way that makes it 'fully prepared' to cope with the challenges in the run. 'Fully prepared' does not only mean accumulating weapons, magic spells and food, but also adding layers of protection against the various dangerous aspects of encountering the level's NPC (i.e. a monster).

Next, three practitioners thought of gameplay as an artefact. To them, this is a set of rules, penalties and awards within a game level, all intended to assert game structure, to control or to influence the players' in-game behavior. As an artefact, gameplay describes the admissible actions, definitions and constraints that apply to the MMOG world. These should be consistent and non-conflicting, and can apply to avatars, NPCs, weapons, spells, and anything else in the game. An interesting dimension of gameplay as an artefact is its linkage to 'the look and feel' document of the game:

"The game designer comes up with the look and feel, a page long description, and then the first cut of the gameplay tells you what it is like to be in the game; you try out the gameplay to get into the looks and feel, and you've got to tell the game designer if the gameplay you experiences is indeed the look and feel he wants in the game." (TechGame)

In the views of our participants, while the Look and Feel document is artistic and expresses a vision for the game, it's the gameplay that is the first "materialization" of this vision. Practitioners thought of it as "*the first prototype of the look and feel*": "*Think of gameplay as the first prototype of your look and feel; you've got to start from somewhere in your game development and here is your starting point.*" (ArtGame)

Last, four practitioners maintained the view of gameplay as a relationship between players and designers, which means endless interaction between game authors and players, in which designers provide the choices, the rules, the penalty and reward scoring schemas, and the players use those as recourses available at their disposal to "*generate their individual gameplay experience*" (as one participant put it). Players, hence, engage in "*writing their individual game story lines*" by using their individual approaches to exploring the choices available in the in-game situations they enter.

Regardless how practitioners conceptualized their understandings of gameplay, they all shared two commonalities:

1. All practitioners were reasoning about gameplay from players' perspective.

"In [MMOG] design, there is only one perspective, and it's the player." (YouGame) From this perspective, gameplay within a level meant providing the player with a set of relevant choices and their consequences. A choice and its consequences are deemed relevant if they are perceived as enjoyable for the player and contribute to the goal of making the player "*want more of the game*". In the views of the practitioners, gameplay is created not only for but also with the players. In this sense, gameplay is co-created where the players are not passive consumers of game contents but also owners of the gameplay as they may decide to use a playing option in a way unanticipated by the game creators. E.g. it's common for players using end game contents to set up their own statistics-processing tools by writing some code with interface to the online game environment, in order to collect and analyze quantitative information showing how a players group progresses in a challenge. In the experiences of our participants, players use this 'in-game analytics' to make their play more efficient: "*Players are informed and concerned about performance and they want to know how far they advanced in killing a monster. 30%? 90%? Imagine you are in a dark cave, too dark that you even do not know where the monster is. You optimize your performance by tracking yours and the monster's statistics.*" (YouGame)

Gameplay co-creation also extends to the real-life world where players deploy off-game analytics in communities of players' web sites and blogs that serve for brainstorming, sharing experiences and crafting strategies for increasing the players' performance to defeat a NPC (i.e. a dangerous creature). For example, all interviewees agreed that their players participate in opinion polls set up outside the game. As per our practitioners, in co-creating gameplay, players also act as source of inspirational ideas for the designers to consider as candidates for inclusion in the next MMOG patch/extension: "*Your players can come up with unimaginable approaches to in-game situations. It's shocking to see players' creativity at play... You need to ask our game masters, the helpdesk guys, who are constantly challenged by players who call to say our help desk line they entered situations in ways unimaginable to the game designers...*" (TechGame)

"Learning from those players [who play the game in unanticipated ways] is a kind of crowdsourcing for

innovation. For example, who could ever have said that players would watch so keenly the avatar's stats and even switch off the features of our beautiful landscape and the sound of the special effects in order to play it more efficiently? I would not tell you the astronomic amount of money this [the sound effects and the landscape effects] all costs!!" (ArtGame)

"Players these days are extremely educated; they do not want just new contents but also some tools to play the game as efficiently as possible. You've got to leverage their ideas somehow..." (YouGame)

Second, all practitioners agreed on the the demarcation of gameplay at lower levels compared to higher levels, and specifically to the endgame levels. In the practitioners' experiences, gameplay applied either to a level in the game, or to the 'end level game contents', but not to the game as a whole. Because the MMOG end goal (namely, continuously improving the fitness of the player's avatar) is perceived as too abstract, practitioners consider it pragmatic to reason about gameplay from the viewpoint of a single game level that has its own (pragmatic) goal – namely, to pass the level, and not from the viewpoint of the whole game. In the words of one interviewee: *"In a business systems project, you break it down in subprojects and you may have very good reasons to determine specific requirements at subproject level. That's how we deal with gameplay. The only way for you to make meaningful specifications of gameplay requirements is within a level."* (ArtGame)

While at lower levels, players may choose individual versus collaborative play mode, for them to be able to cope with the challenges at the endgame levels, they must join a team of players and only collectively they tackle the challenges. In fact, many players go through the levels to arrive as fast as possible to the end level which "*offers a completely different gameplay*" in the sense that its focus is completely on collaboration. Ten out of 12 practitioners referred to this as to the so-called "*group-or-die*" gameplay because the only way for players to continue their play was by joining a group (or raiding guild, in the terminology of the game community).

B. Requirements Artefacts Dealing with Gameplay

All practitioners thought gameplay requirements form a component ("chapter", "essential part with its own heading") of the game design document, the pillar of every game development project. It is a living document, in the sense that it's gradually refined following agile project management philosophy. When practitioners refer to it as 'complete', they do not mean 'carved in stone' or frozen, but 'complete' in the sense that it's enough for developing a piece of playable functionality. Similarly to the game design document, the gameplay requirements document component is also a living artefact. Its first version is grounded on the game's Look and Feel (LaF) document which in MMOG projects has the meaning that the project mission and vision documents have in conventional system development project (e.g. enterprise systems). Each gameplay requirement change is checked for alignment against the statements in the LaF document,

meaning that gameplay should support - at all times, the look and feel of the game as envisioned by the game designer. As gameplay requirements are elaborated gradually, traceability links are maintained between the gameplay changes and the affected other parts of the game design. The gameplay requirements document consists of six parts: (1) wireframe, (2) rules of the game, (3) playing modes, (4) scoring system, (5) level specifications, and (6) a process flowchart. The wireframe is the visual blueprint that arranges the game contents for the purpose of achieving the look and feel specified in the LaF document. It can be thought of as "the map of the interfaces". The concept of rules is self-explanatory (the meaning has already been described in the previous section). The scoring system is the total of awards and penalties. Furthermore, levels specifications explicitly state the design requirements specific to each level in the game. Last, the exact process that the player has to go through within a level is specified by means of a flowchart. The format of the flowchart can vary, for example can be manually made cartoon-like pictures, simple flowcharts based on the built-in Visio stencils, or sophisticated process diagrams using process modelling tools (surprisingly, one interviewer used a business process modelling tool that his company also deployed for documenting business process workflows in accounting and human resource management). Each level's flowchart is detailed into features. For example, each MMOG in our case study had a proprietary editor (e.g. a map editor, a layout editor) and its features were listed in detail.

Another distinctive artefact that we observed to depend on gameplay requirements was the so-called 'paper-prototype'. It is specific to prototyping gameplay requirements at lower game levels (when no code is yet developed). It can take the form of sketches, mock-ups, pencil drawings or more complex artistic drawings. It is therefore a low-cost and easy-to-make way to demonstrate gameplay to the producing company's employees as well as relevant stakeholders. Gameplay is then experienced and feedback by those who experienced it is collected to confirm gameplay requirements. In all the companies, the colleagues of the professionals who made the paper prototype were the first to "*play with it*" in order to experience "*how it feels to be in the game*". Of course, the very first players to whom a prototype is shown are the team members in the game development project: *"It could be everyone that is at work that day; you can ask your artistic director to be a player, or your special effects artist, or a junior or a senior programmer."* (ArtGame)

At higher game levels that feature increasingly more community-based play, the paper-prototype is usually followed by developing a software prototype, an artefact used to demonstrate how playable are the gameplay requirements from players' community perspective. The practitioners deemed this artefact central to gameplay requirements validation as it meant collecting and analyzing players' feedback (at community level).

C. How Our Study Participants Manage Gameplay RE?

All practitioners perceived the gameplay RE process as creative, iterative and non-linear one, in which professionals

must be open-minded and be prepared to alternate “*fast track lanes with scenic routes*” (*TechGame*). An interesting observation is that no practitioner was able to describe it as a predefined flow of steps that he/she attempted to use in his/her projects. Some interviewees shared that to an external observer this process usually appears as “*chaotic*”, “*judgment-based*”, “*gut-feel-grounded*”. They all meant it as something that does not lend itself easily to a “*process description as you do in business process reengineering projects*” (*YouGame*). Instead, they thought of it rather as a ‘system of spaces’ in which spaces delimit various kinds of related activities that together form the continuum of gameplay RE. They referred to them as ‘spaces’ and not process phases or activities, because ‘the spaces’ did not make up any linear structure nor were executed sequentially. While carrying out the activities might look chaotic, in the views of the participants the process consistently brought the targeted results in their projects: “*I've never seen a linear, milestone-based process here. We do have a process architecture for gameplay requirements but it's totally different, just like out of this world... It's like you have a bunch of tools in your toolbox but without knowing your game level, and its feel and look, you have no way to guess what tool to pull out first; even if you do know the look and feel, and you know the right tool for it, you may find out that it does not fit right in, so you've got to tweak it a bit and if it would not work, then go pick another tool. You should try out different things and see what works and then make sure I do more of what works and less of what does not...*” (*YouGame*)

A common concern across all the spaces was the ‘balancing of the gameplay requirements’. To practitioners, this meant getting a requirements definition that leads to creating a playable game version that: (1) is fair with respect to all players involved, (2) brings more pleasure to play as player’s learning and experience advances, and (3) is free of repetitive in-game options, which means that no playing option is worthless and the net value of each option is commensurate with the cost of using it. Our findings suggest that balancing the gameplay requirements is recursive in nature because achieving balanced gameplay with respect to one of the three criteria above may cause an imbalance to emerge with respect to another criterion and this imbalance may need to be smoothed first, before the first one would. As one interviewee said: “*You want to get the gameplay crispy like chips. But chips are fragile, and you'd better treat them that way. The crispier my gameplay gets in one dimension, the more fragile it might get on another, and if this happens then we'd go first to fix the second issue and once is done, then get back to the first. You get back and forth many times until you make it fun to play; fun AND fair, I mean.*” (*ArtGame*)

To cope with gameplay requirements in a way that gameplay balance is achieved, practitioners experienced that they were going through three spaces (Fig. 1). We labelled the spaces ‘prototyping’, for the activities that lead to creation of paper prototypes or software prototypes, ‘play-testing’ for activities centered around the discovery of emergent requirements, and ‘tweaking’ for the gameplay redesigning

activities that lead to improving the players’ path through the game levels. We note that prototyping and play-testing have the meaning of player-centric exploration and are expected to help discover new requirements, as in MMOG projects most requirements emerge throughout the game design itself. In contrast, tweaking has the meaning of an intense synthesis of all learning that happened in prototyping and play-testing and use of this synthesis to redesign the gameplay.

We observed however that it was hard for practitioners to pin-point to one particular space that is more important than the others to the balancing of the gameplay requirements. Instead, they considered all three spaces to be integral to it, and deemed iterations to be the glue that sticks them together.

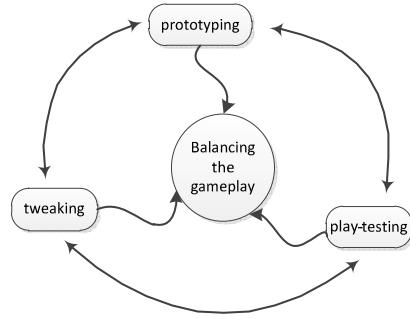


FIGURE 1. RE SPACES USED TO BALANCE GAMEPLAY

E.g., in her most recent project the playability analyst at *TechGame* used prototyping and play-testing as spaces for exploring various amounts of story-telling and full motion video surrounding an end level NPC and a players’ guild. Used together, these two spaces led to the creation of up to five new alternative game designs for segments of the game environment (tweaking). Each tweaked design treated differently the players’ proactiveness and needs of control. The designs also varied in terms of availability, ease to use and players’ cost of the game features included, and these three had to be balanced against their value for the players. The designs were subjected to prototyping and play-testing again whereby prototyping and play-testing were in different proportions regarding each alternative design. These alternating ways of using spaces were continued until the game would be deemed ‘balanced’ by not only its designers but also by its players (in the targeted community).

V. DISCUSSION

This section puts our finding in the perspectives addressed in prior publications in RE for game systems as well as in other fields (e.g. human-computer interaction, game design). Our discussion follows the topics in the previous section.

Gameplay and Gameplay Requirements. We found three ways to think of gameplay. While this diversity may be traceable to the various background of the people involved in the multidisciplinary teams in MMOG development, we think it also reflects the lack of standard definitions of terms in the gaming domain and, in turn, the lack of common understanding in the various communities regarding what gameplay means. This is consistent with RE textbooks (e.g.

[27]) pointing out that such a challenge is typical for new domains that call for multidisciplinary approaches to solution development. Next, our results suggest that gameplay requirements are perceived as functional requirements and not as non-functional ones (quality attributes). They are mapped out by using flowcharts or lists of features [27] which are common techniques for specifying process aspects in functional requirements. We also found non-functional aspects pertaining to gameplay to be part of the LaF document where playability and fun are explicitly mentioned as quality attributes. We did not explore the dependencies between the LaF document in the light of functional and non-functional requirements that make up gameplay. Nor we explored how the relationship between playability and gameplay. We thought think that such an exploration is a necessary step in order to gain complete knowledge of how three concepts (fun, gameplay and playability) relate to each other. We, therefore, suggest it as a line for future research.

Also, our observations suggest the concept of goal plays a prominent role in reasoning about gameplay requirements. We found, gameplay is bound to a level and as such - to a clearly specified goal. This may mean that gameplay requirements would naturally lend themselves to the goal-oriented RE (GORE) approaches. Our practitioners however have shown no awareness of the deployment of these approaches in the MMOG industry. As part of this study, we made a search in the Scopus database (www.scopus.com) for empirical studies that used GORE in game context. We could not obtain any hint about any attempts to deploy any RE method resting on the goal-oriented paradigm. We therefore think that empirical work to evaluate the fit of GORE and MMOG projects is necessary and worthwhile because it potentially could reveal how existing RE methods could be leveraged to this new domain. Such studies could inform both practitioners and researches on the benefits and limitations of GORE in this particular context.

Gameplay artefacts. Our study suggests the gameplay requirements take a prominent position in the game design document. This agrees with the studies [13,14] regarding the fact that gameplay and game design are linked concepts. However, while [13] presents gameplay as an artefact on its own, preceding the elaboration of the game design, our study found that (1) the LaF document precedes the process of gameplay requirements development and (2) that the gameplay requirements do not precede the game design, but are an integral part of the game design. In other words, gameplay requirements and game design are co-evolving together. Our observations that gameplay requirements form living documents and are created by using agile development philosophy agree with textbooks in game design (e.g. [3]).

Perhaps the most important finding in the study is the joint co-creation of gameplay requirements by game designers and players (organized in communities, or guilds). While the designers create the choices and the consequences for players taking these choices, the players are inventive in the sense that they (1) create rules of play that are not intended by the designers, (2) instill codes of conduct, (3) impact how the

game is facilitated, and (4) impact how development plans would go next. This finding agrees with the study in [11] that found gameplay as socially negotiable, which means that players and game developers together determine the rules. One could also think of parallels in meanings between our concept of gameplay co-creation and the concept of player-managed gameplay identified in the survey-based study of Deaker et al. [8] of the World of Warcraft, a leader in the MMOG market.

Our finding that gameplay is co-created with players using in-game and out-of-game analytics, could be explained with the proactiveness of the players and their willingness to form groups. As in [33], the end level gameplay experience (of World of Warcraft) is produced through guild-centric collaboration, exclusively. Furthermore, our results make us think that the intensity of co-creating gameplay seems to grow with the advancement of the levels at which players plays. Most blogs and in-game and out-of-the game analytics tools were known to be created by players identifying themselves with the end level of the respective MMOG, included in our study. We however consider this a working hypotheses only and future research is needed to collect evidence that confirms or disconfirms it.

Our Practitioners' Managing Gameplay RE. Our findings suggest no one-size-fits-all process to cope with gameplay RE. Instead, the case study companies had spaces that collectively helped deliver the requirements. We note that we preferred the concept 'spaces' over 'steps' because in the experiences of the practitioners they usually were not done sequentially. E.g., gameplay requirements may loop back through prototyping, tweaking, and play-testing more than once as the game development team refines them and explores new directions. This agrees with [38] regarding the role of prototyping. However, our finding seems to disagree with [13] that prototyping is hard. In fact, we found that the paper-prototype was a high value/low cost way to cope with gameplay requirements.

Our understanding of managing gameplay RE by using spaces is in line with the creative processes studied in the field of design, e.g. [28]. We think this is unsurprising as gameplay requirements definitions result out of the creativity of contributors bringing a broad range of backgrounds. We however notice that no RE research has applied so far the theoretical lens of design thinking to the study of RE phenomena in the gaming industry. For this reason, we think it would be interesting to investigate the commonalities between managing gameplay requirements and design thinking approaches [28]. This forms a line for future research.

VI. VALIDITY THREATS

We used the checklist in [30] to assess the possible threats to validity in this study. As it is exploratory, the key question to address when evaluating the validity of our results, is [22]: to what extent can the practitioners' experiences in gameplay RE be considered representative for other MMOG companies and other (non-MMOG) game producers? While the case study companies are not representative settings for all the possible ways in which gameplay RE can happen, following [31] we

could think that it might be possible to observe similar experiences in MMOG projects and companies which have contexts similar to those in our study. E.g. other MMOG producers who use the subscriptions-based business model and have incentives for designing gameplay that serves the purpose of player retention, and has no end goal to achieve. This subscription model requires companies to commit to high quality standards (so that they retain their player base as long as possible) and incentivizes them to come up with steady releases of content. Such MMOG organizations might experience observations similar to ours. Would our findings hold for other types of games? We think that as the concept of gameplay is essential to any type of gaming systems, it might well be possible that a part of our findings would be observable in the contexts of any game that is structured in levels, e.g. gameplay as an artifact (a set of rules, penalties and awards) is likely to apply to any game. However, we think that it's more likely to observe findings similar to ours in projects delivering games for which the only purpose is enjoyment (like e.g. in [38]). If we replicate this study in serious games contexts, the results would be different since pedagogical goals and learners' behavior models dominate the game design and impact the choices and the time available to players.

There are also three inherent weaknesses of interview techniques [23] that we accounted for in this study. First is the extent to which the participants answered our question truthfully. We attempted to minimize this threat by involving volunteers, under the assumption that if a practitioner would not be able to be honest, he/she could decline his/her participation at any stage of the research. However, in interview studies there is always a residual risk that the respondents were extremely selective in choosing the documents for sharing with the researcher and that the respondents might have incentives to share the very best of their work. Second, it is possible that the researcher might instill her bias in the data collection process. We followed Yin's recommendations [22] in this respect: (i) we included participants with diverse jobs and with diverse contributions to the gameplay RE process; this allowed the same phenomenon to be evaluated from diverse perspectives (data triangulation [23]); (ii) the interview answers were sent to each participant prior to data analysis to confirm the information he/she provided; and (iii) we had the data analysis report reviewed by some interviewees (some of whom provided feedback on clarifying the concepts, but this did not change the analysis).

VII. CONCLUSIONS AND IMPLICATIONS

This exploratory study explicates what practitioners in MMOG development teams think of gameplay requirements and how they manage gameplay RE. It found that gameplay requirements have multileveled meanings. This finding has some implications which seem to make it hard for companies to come up with a common process for gameplay RE. Instead of RE processes, we found that practitioners think in terms of RE spaces (and not steps) and these spaces collectively help addressing the key concern in MMOG: balancing the gameplay. Our study has the following implications for RE

researchers: First, in section V. we presented a number of possible lines for future research. We observed that our findings in gameplay RE did not call for development of new methods. Instead, knowing what we know now, it seems to us that more evaluation work would be an important thing to do first. For example, as we observed that the concept of goal is associated with a game level, and not with the game as a whole, we think that gameplay requirements within a level might be possible to approach by using GORE techniques. The evaluation of the applicability of GORE-methods to MMOG projects seems therefore relevant. Also, the developing game pattern community in the field of game design suggests ontology-based techniques (e.g. [5, 7] that could be adapted to serve RE purposes. Empirical evaluation on how these approaches can be leveraged in MMOG context is therefore worthwhile. Last, the three ways to conceptualize gameplay could be subjected to a more comprehensive survey with participants from a large number of companies. If confirmed, our findings would have broader implications to RE (e.g. we may need to design RE tools that support this multiple conceptualization view).

Second, this study implies some methodological considerations: (1) Our results suggests that gameplay is a concept with layered meanings that could lend themselves to multidisciplinary empirical research in which researchers could possibly use existing theories from other disciplines, e.g. psychology, media communication, to explain gameplay in more depth. E.g. our study indicated the joint co-creation of gameplay requirements by game designers and players in the MMOG domain. To understand the internal working behind the relationship between designers and players in RE, we think that future research could leverage theories about client-vendor value co-creation (e.g. [45]) from the area of e-commerce. MMOG are examples of e-commerce and therefore would be a suitable area for the application of these theories for the purpose of understanding gameplay requirements. (2) The MMOG world of players interacting online gets increasingly more distant from the traditional lab settings we know in RE-experiments. As MMOG environments keep track on a broad range of statistics and each player generates huge volumes of data all being recorded in the game system, an empirical approach to investigating gameplay as it happens real-time, could be the direct observation method. A researcher might consider entering the game, creating his/her own avatar and then collecting observations of how social gameplay works, while playing the game. E.g. a researcher can evaluate theoretical models of coordination and collaboration in the context of social networking in online game communities. Examples of this research approach are provided in [9] and [32]. Leveraging these experiences in the RE field is a line for future research.

This study has some implications to RE practice. To RE specialists who look for opportunities to join the game industry, our research indicated that the players' perspective is the one driving gameplay RE. To get intimate knowledge of the players' perspective, it seems unavoidable for RE specialists (who want to switch to the game industry) to become MMOG

players first. In contrast to other domains, e.g. accounting, health-care, where RE professionals do not have to be actual users of the systems to be able to understand the world of the users sufficiently well and do decent job, this study revealed that being a player seems a prerequisite for being a good gameplay requirements engineer. In a world where the product (the game) is produced by people with drastically diverse background, as Callele et al put it [13], being a passionate game-player may well be the only commonality that sticks professionals together.

To RE tool vendors, our findings that gameplay requirements are perceived as functional requirements and that traditional flowchart and process modeling techniques are a good fit with gameplay modelling mean that there could be a new market segment just waiting to be explored. To the best of our knowledge no RE tool vendor visible in the RE community has geared their tool to serve MMOG companies. This study suggests that MMOG companies could become potential users of such tools, i.e. they form a market and maybe tool vendors might be interested in exploring and developing it further.

ACKNOWLEDGEMENT

The author is indebted to the IGDA representative and the participants for taking time out of their busy schedules to join this study. Without their sincere response and enthusiasm this study would have been impossible.

REFERENCES

- [1] DFC Intelligence, Online Game Market Forecasts 2014 (2014, Feb), San Diego, USA.
- [2] De Heij, B., et al, (2013). The Global Games Market: Newzoo Trent Report, Newzoo Inc., Green Bay, USA.
- [3] Fullerton T. (2008). Game Design Workshop: A Playcentric Approach to Creating Innovative Games, CRC Press, USA.
- [4] LeBlanc M. (2006). Tools for Creating Dramatic Game Dynamics. In: Salen, K. & Zimmerman, E. (eds.) The Game Design Reader: Rules for Play Anthology, MIT Press, 438-459.
- [5] Zagal, P.J., Bruckman, A. (2008). The game ontology project: supporting learning while contributing authentically to game studies. ICLS (2), 499-506
- [6] Bjoerk, S., Holopainen J (2005). Patterns in Game Design, CRM.
- [7] Falstein, N., (2004) The 400 project/ Retrieved March 1, 2014 from https://www.theinspiracy.com/400_project.htm
- [8] Deaker, C., et al, (2012). A Survey of Players' Opinions on interface Customization in World of Warcraft, ACE, 198-213
- [9] Gross, S., et al, (2012). Studying Social Relations in MMOG Play: an Illustration of Using Ethnography to Frame "Big Data", 17th ICCG, 167-174
- [10] Bergstroem, K., et al, (2010). Exploring aesthetical gameplay design patterns, MindTREK 17-24.
- [11] Nardi B. (2010). My Life as a Night Elf Priest, Ann Arbor: Univ. of Michigan Press.
- [12] Brockmyer J. H., et al, (2009). The development of the Game Engagement Questionnaire: A measure of engagement in video game-playing, J of Exp. Social Psychology, 45(4), 624-634
- [13] Callele, D., et al, (2005). Requirements Engineering and the Creative Process in the Video Game Industry. RE'05: 240-252
- [14] Alves, C.F., et al, (2007). Challenges in Requirements Engineering for Mobile Games Development: The Meantime Case Study, RE, 275-280
- [15] Callele, D., et al, (2009). Augmenting Emotional Requirements with Emotion Markers and Emotion Prototypes. RE, 373-374
- [16] Draper, S. (1999). Analysing Fun as a Candidate Software Requirement, Personal technology, 3(1), 1-6
- [17] Hasselzahl, M., et al, (2001). Engineering Joy, IEEE Software, 18(1), 70-76
- [18] Bentley, T., et al, (2005). Putting Some Emotions into Requirements Engineering, 7th AWRE.
- [19] Ampatzoglou, A., Stamelos, I. (2010). Software engineering research for computer games: A systematic review. IST 52(9): 888-901
- [20] Cornett, S. (2004).The Usability of MMOG: designing for new users, SIGCHI Human Factors in Comp. Syst, ACM, 703-710
- [21] Gonzales S.J.L., et al (2013). Playability: analyzing user experience in video games, Behaviour & IT 31(10), 1033-1054.
- [22] Yin, R.K. (2008).Case Study Research, Sage.
- [23] King, N., Horrock, C. (2010). Interviews in Qualitative Research. Sage.
- [24] Bensabat, I., et al, (1987) The Case Research Strategy in Studies of Information Systems, MISQ, 11(3), 369-386.
- [25] Charmaz, K. (2007). Constructing Grounded Theory, Sage.
- [26] Daneva, M., et al (2013) Agile requirements prioritization in large-scale outsourced system projects: An empirical study. JSS 86(5): 1333-1353
- [27] Lauesen, S. (2002). Software requirements: Styles and techniques, Addison-Wesley.
- [28] Brown T., Design Thinking, HB Review, June 2008, 84-95
- [29] Pereira L.L., Roque, L., A (2013) Preliminary Evaluation of a Participation-centered Gameplay Experience Design Model, SouthCHI, 332-348.
- [30] Runeson P., Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. ESE 14(2), pp 131-164.
- [31] Seddon P., Scheepers, P. (2011). Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples, EJIS, 1-16.
- [32] Castranova, E., et al, (2013). Designer, Analyst, Tinker: How Game Analytics Will Contribute to Science. Game Analytics, Maximizing the Value of Player Data. Springer, 665-687
- [33] Ducheneaut, N., et al, (2006). Building an MMO with Mass Appeal: a Look at Gameplay in World of Warcraft, Games and Culture 1(4), 281-317
- [34] Dickey, M. D. (2011). World of Warcraft and the impact of game culture and play in an undergraduate game design course, Computers & Education 56, 200-209
- [35] Prahalad, C.K, Ramaswami (2004). The Future of Competition: Co-Creating Unique Value With Customers, HBR Press.
- [36] Pasquale, L., et al, (2013). Requirements engineering meets physiotherapy: An experience with motion-based games, REFSQ, 315-330
- [37] Cooper, K.M.L. et al, (2014). Towards model-driven requirements engineering for serious educational games: Informal, semi-formal, and formal models, REFSQ, 17-22.
- [38] Kasurinen, et al, (2014). Is Requirements Engineering Useless in Game Development? REFSQ, 1-16

Therapist-Centered Requirements:

A Multi-method Approach of Requirement Gathering to Support Rehabilitation Gaming

Cynthia Putnam and Jinghui Cheng

College of Computing and Digital Media

DePaul University

Chicago, IL, 60604, USA

cyputnam@cdm.depaul.edu, jinghuicheng@gmail.com

Abstract—Brain injuries (BI) are recognized as a major public health issue. Many therapists include commercial motion-based videogames in their therapy sessions to help make rehabilitation exercises fun and engaging. Our initial exploratory work exposed a need for tools to help therapists make evidence-based decisions when choosing commercial motion-games for their patients who have had a BI. Targeting this need, we are gathering requirements for a case-based recommender (CBR) system that will act as a decision tool for therapists. In this paper, we describe our ongoing work as a case study that illustrates our multi-method approach of requirement elicitation for the CBR system. Our approach is comprised of four overlapping steps: (1) interviews with therapists, (2) onsite observations of therapy game sessions, (3) diary studies in which therapists record detailed information about game sessions, and (4) a user study of a CBR prototype interface. Leveraging direct interaction with end users (i.e., therapists), this case study demonstrates requirements gathering techniques to address needs of a special population (i.e., therapists who work with patients who had BIs) in a specialized context (i.e., inpatient rehabilitation using motion-based video games).

Index Terms—Requirements elicitation, Multi-methods, Brain Injury, Rehabilitation, Therapy, Games, Case-based Recommender

I. INTRODUCTION

The Center for Disease Control and Prevention (CDC) in the US recognizes that brain injuries are a major public health issue [1]. Approximately 6.4 million children and adults in the US live with a lifelong disability as a result of a traumatic brain injury (e.g. falls, car accidents) or a stroke [2]. People who have experienced a BI exhibit a wide range of physical and cognitive disabilities. Physical disabilities can affect both gross and fine motor coordination [3]. Full or partial paralysis is common especially with stroke, often affecting speech [4]. Patients often exhibit gait and balance impairments, which also increase the risk of falling and encountering further injury [5]. Cognitive disabilities (e.g., impaired memory, reasoning and problem solving abilities) are also common resulting in many day-to-day difficulties including problems in following directions (way-finding), completing procedural tasks (shortened attention spans), and understanding language [5]. Due to the wide-ranging causes, effects and recovery arcs associated with BIs, rehabilitation treatments vary widely and *need to be customized by therapists for each patient*.

Clinical experience and research has found that it is often challenging to motivate people with BIs to perform the repetitive exercises commonly prescribed for rehabilitation [6-8]. To address this challenge, many therapists include commercial motion-based videogames in their therapy sessions to help make repetitive actions fun and engaging. (Commercial motion-based videogames as those that use consoles; e.g., XBox Kinect, Nintendo Wii and Sony Move, which are played using gestures and/or sometimes-specialized controllers.) Commercial motion-games have been found as effective motivators for performing rehabilitation exercises [9]. Moreover, they have been found effective in helping people with BIs meet their therapeutic goals, including improvements in balance and range of upper extremity motion [10-11]. While this work is encouraging, it has focused on relatively small samples; as such, very little is understood about the mix of attributes (game, patient, therapy session) across the diversity of people with BIs that need to be considered for successful use of commercial motion-gaming in BI therapies. In other words, there is a paucity of information to help therapists consider customized treatments that include the use of commercial motion-games for the wide range of people with BIs that they treat.

In this paper, we describe our ongoing work as a case study to illustrate our four-step approach of requirement elicitation for a software system. The system will help therapists make evidence-based and customized decisions when choosing commercial motion-based videogames for their patients who have had a BI. Our goals of this study are to (1) explore the problem domain of therapeutic gaming in BI rehabilitation, (2) identify the therapists' requirements for choosing games for BI rehabilitation, and (3) identify specific requirements for attributes that contribute to good or bad game choices in the context of a specific patient and therapy session. This work contributes to discussions in requirements elicitation on methods that include users to elicit system requirements (e.g., [12-14]). Because of the challenge of selecting appropriate elicitation techniques, several researchers have proposed frameworks for deciding which methods to use (e.g., [15,16]). Additionally, researchers have also discussed traditional methods in requirement gathering; i.e. the use of interviews (e.g., [17]), observations (e.g., [18]), and ethnographic techniques (e.g., [19,20]). While leveraging traditional methods (i.e., interviews and observations), we also illustrate the use of

a non-traditional method, i.e., diaries, for requirement elicitation within a multi-method framework. We anticipate that this case study will serve as a model of requirements elicitation in a specialized context.

Our envisioned system will also serve as a tool to support game developers and designers that want to design therapy-centered games; in other words, it will help further requirements gathering about games customized specifically for use in BI rehabilitation. This aspect of the project has overlap with research focused on requirements analysis to better address a wide range of user needs [21]. For example, Sutcliffe, Fickas and Sohlberg (2005) discussed the need for personal and contextual requirements to address diverse users, including people with disabilities. Specifically, in their framework, user *characteristics* and personal *goals* are considered as additive layers applied over functional specifications. This layering affords individual customization of systems to address diverse users' needs [22]. Our envisioned software system uses Sutcliffe et al.'s framework conceptually to consider patient *characteristics* and therapeutic *goals* to facilitate the understanding of personalized patients' needs when using motion-based video games.

II. OUR INITIAL EXPLORATORY WORK

To explore the problem domain of therapeutic gaming in BI rehabilitation, we started this project by conducting exploratory interviews with therapists who used commercial games with their patients that had BIs. The exploratory interviews were conducted in June-October of 2012 with four physical therapists, two occupational therapists, and three recreational therapists, and two speech-language pathologists ($N = 11$). All therapists worked at the Schwab Rehabilitation Hospital in Chicago, Illinois, with people who have had BIs. As part of the interviews we asked therapists how they chose motion-games for their therapy sessions and what games they used. Therapists identified familiarity (i.e., games they already knew about) as their primary reason to choose a game. Moreover, despite the proliferating motion-game market, the therapists were using a very limited number of motion-game titles. In other words, there was a perceivable information deficit among therapists that we interviewed about options available in commercially available games.

Following the interviews we installed AV carts that housed the three major gaming consoles with games the therapists selected; we then observed sixteen motion-gaming therapy sessions. We observed that to maximize the limited 45-minute session times common at Schwab, it was critical to know if a selected game was going to be (a) a good match for the patient's play preferences and abilities, and (b) address the therapeutic goals for the session. We witnessed several sessions where the games were not a good match. In these sessions, patients were generally frustrated or bored and often verbalized negative feelings about themselves. Further, while therapists had specific goals in mind for the play sessions (e.g., weight shifting, balance), they had no ability to choose games based on evidence to support their efficacy at addressing those goals.

In sum, the exploratory work exposed a need for therapists to have access to information about games (e.g., those that best met their therapeutic goals and the abilities of their patients) that would help them make evidence-based decisions; *in other words, help in customizing the choice of commercial motion games for their patients with BIs*. To address this need, we are gathering user and system requirements for a case-based recommender (CBR) system that will act as a decision and information tool for therapists. CBR systems solve problems by referencing previous solutions or 'cases' [23]. They have been used in a wide range of fields, including many medical applications [24]. For example, Ahmed and Funk (2012) created a CBR system to help physicians choose a post-operative pain treatment plan for their patients [25].

III. FOUR STEP APPROACH: METHODS AND FINDINGS

Working with therapists at Schwab and Marianjoy Rehabilitation Hospitals (Wheaton, Illinois), we are using a multi-method approach of requirement elicitation for the CBR system. This approach has involved four overlapping steps: (1) interviews, (2) onsite observations, (3) diary studies, and (4) user studies of a prototype CBR interface. Each method has elicited different aspects of system and user requirements that are critical for the CBR system design:

- The interviews with therapists have helped to establish what they will need to input into the CBR system to help make evidence-based decisions including: (a) session goals, and (b) patient factors, e.g., patient abilities and personal preferences.
- The observations have established contextual factors the CBR system will need to consider when providing recommendations, including patient scaffolding issues and time limitations.
- Diary studies clarified the requirements by helping us define the attributes in a 'case' and have provided 'seed cases' that determine how well (from therapists' perspectives) the motion-games matched the session goals and patient factors.
- The user studies of the system interface have helped us understand required interface and content considerations, including (a) input controls and (b) output display.

In the following sections, we describe each method step in greater detail. We are only reporting here on findings that affect the CBR system design, i.e., many of the findings that had implications for game design are not included here.

A. Step One: Interviews

While the initial interviews with therapists at Schwab led to the need for a CBR system, the interviews (at Schwab and later at Marianjoy) also helped to establish general therapists' needs that are important to consider for a CBR system.

1) *Participants*: Among the 21 therapists we interviewed at the two rehabilitation hospitals, five were occupational therapists, nine were physical therapists, three were recreational therapists, and four were speech-language pathologists. At the time of the interviews, all therapists were

working in inpatient care with people who had BIs; therapists had between two and fourteen years of experience.

2) *Procedures*: Therapists were recruited through an email sent from their Quality and Research offices; as such, they represented a convenience sample. We conducted the interviews in June and September of 2012 at the Schwab and in September 2013 at Marianjoy. Interviews took 30-45 minutes; all were video recorded and later transcribed. Therapists were asked several questions about how they used motion-games. The questions we asked affecting CBR system design include: (1) major therapeutic goals they targeted; (2) major motivations for using motion-games; and (3) factors they considered when choosing games.

After the interviews were transcribed two researchers on our team independently analyzed the interviews and inductively coded for salient themes; they then co-wrote a codebook on how to identify the themes. (We are using ‘code/coding’ here as it is referred to in qualitative data analysis: “*A code in qualitative inquiry is most often a word or short phrase that symbolically assigns a summative, salient, essence-capturing, and/or evocative attribute for a portion of language-based or visual data*” [26].) A third researcher used the codebook to deductively analyze the interviews using the codebook. Pairwise inter-rater reliability between the third coder and the first two was calculated using Cohen’s Kappa. For all findings described in the next section, average pairwise reliability was between .71 or and .88 (.61-.80 is considered ‘good’ and above .81 is considered ‘very good’ [27].)

3) *Summary Findings Affecting CBR System Design*: We focused on (a) goals, (b) motivations and (c) factors affecting game choice/use for the CBR system design.

a) *Goals*: While the speech-language pathologists only spoke of cognitive-related goals, all 21 therapists mentioned cognitive goals. The top three cognitive-related goals were (1) sequencing, (2) focusing attention, and (3) motor planning. However, the occupational, recreational and physical therapists were more concerned with physical goals. The top three mentioned were (1) balance, (2) weight shifting, and (3) standing endurance. For example, Interviewee 07 (occupational therapist), said, “***Traditionally the biggest thing that I used it for is when I'm working with patients with balance, especially when I want to work on the timing component of that.***”

b) *Common motivations*: The most common motivations for using motion games in therapies was to (1) add fun, (2) introduce something novel, followed by (3) a desire to distract patients. In an example of all three motivations, Interviewee 02 (physical therapist) told us, “***It gets kind of boring and we try to think outside the box and make things fun and exciting for patients so they have fun during therapy and it's not just the same old rote exercise or everyday things that they would not find motivating. I think that games are motivating. I've found that people forget how hard they are working while they are caught up in a game.***” Therapists who were involved with group gaming sessions also emphasized the power of games to encourage social interaction.

c) *Patient factors that affected game use*: The top factor mentioned were cognitive abilities; therapists felt that commercial games often had too many distractions (e.g., noise) and required too much sequencing to be tenable for many people with BIs. Patients’ age and game experience were also considered very important inter-related factors; specifically, therapists were much more likely to use games with younger patients who had past gaming experience. For example, when asked about whom she used motion games with, Interviewee 01 (occupational therapist) responded, “***Typically it's the younger - adolescent to young adult that has used games before.***”

B. Step Two: Observations

To further understand contextually situated requirements for the CBR system, we conducted onsite observation of therapy sessions in which therapists use motion-games with their patients. After the interviews, we installed AV carts in both rehabilitation hospitals that housed the three major gaming consoles (Nintendo Wii, Xbox with Kinect, and Sony PlayStation with Move) with games the participating therapists selected, see Fig 1.



Fig 1. AV cart set-up at SRH

The games therapists selected fell into five categories: (1) sports, e.g. ‘Kinect Sports’, ‘Wii Sports Resort’, ‘Sports Champions 2’ for the Sony PlayStation; (2) fitness, e.g. ‘Wii fit’, ‘Your Shape Fitness Evolved’; (3) dance, e.g., ‘Just Dance 2’, ‘Michael Jackson The Experience’; (4) games with general physical activities, e.g., ‘Kinect Adventures’, ‘Minute to Win it’; and (5) games focused on cognitive skills but that required motion-based controls, e.g., ‘Brain Body Connection’. While the initial observations at Schwab led, in part, to the need for a CBR system, the observations have also provided contextual factors the CBR system must consider. While similar, the structure of care was slightly different between the two hospitals:

- *Structure at Schwab* - Patients were scheduled for up to four 45-minute sessions per day. Therapists worked with up to eight patients a day’ patients were scheduled with physical and occupational therapists and speech-language pathologist who in turn would work with recreational therapists if they felt that their patient would benefit from recreational therapy. Additionally, there were three social/gaming sessions a week.

- *Structure at Marianjoy* - Patients were under the supervision of physical therapists and were scheduled for at least four hours of therapy in 30 or 60 minute sessions. The supervising physical therapist would schedule sessions with occupational therapists and speech-language pathologists based on patient need. Marianjoy did not employ recreational therapists. Seven of our observations were 30-minute sessions; one was an hour-long session.

1) *Participants*: We observed sixteen game play therapy sessions in July through October of 2012 at Schwab, and eight in October 2013 at Marianjoy. Observations were video recorded from the front (for facial expressions) and back. Patients ranged in age from 34-69; all had survived their brain injuries within thirty days of our observations. Cause of brain injury was diverse, including stroke, falling accidents, and seizures associated with alcoholism.

2) *Procedures*: Prior to the observations, therapists completed a “Documentation of Capacity”, that stated that the patient was capable of reading, understanding, and signing our consent forms. As such, we were limited to observations in which patient participants were not highly cognitively affected by their brain injuries.

Qualitative coding of video recordings has been established as an effective way of analyzing video data [28]. We coded the Marianjoy video recordings in 10-second increments to identify: (1) patient’s activity (e.g., playing, resting, and seeking help); (2) therapist’s activity (e.g., providing physical and cognitive support, setting up the game); and (3) patient affect (e.g., happy, focused, frustrated). (*We did not code the Schwab videos.*)

3) *Summary Findings Affecting CBR System Design*: Typically, patients played one to three games in a session. The most common game categories used in descending order were: (1) fitness, specifically the Nintendo Wii Fit mini-games in which the patient stood on a platform that sensed weight-shifting, balance, and foot movement; (2) sports, specifically Wii and Microsoft Kinect bowling; and (3) general gaming requiring physical action, specifically Kinect Adventures. Four findings from the observations have implications for the CBR system design: (a) therapists mobility, (b) patient scaffolding, (c) assistive devices and (d) timing.

a) *Therapists mobility*: Therapists in both locations spend most of their day in the therapy gym. Marianjoy had computers on mobile carts and at the sides of the gym; Schwab therapists had a cubicle space in offices off the gym. As a result, the CBR system needs to be accessible from both PCs and mobile devices (smart phones or tablets), which therapists can use while working with patients in the gym. The major implication for the CBR system design was consideration for a mobile first design [29].

b) *Patient scaffolding*: We found that even the high functioning patients we could observe needed a lot of cognitive and physical scaffolding; for example, during gameplay therapists often had one or more hands on the patient’s waist or used a supportive belt. In the five of the

eight Marianjoy sessions that we coded, therapists were providing physical support 100% of the time the patient was playing a game. For the remaining three session physical support ranged between 3% and 86% of the patient playtime. Therapists also spent an average of 42% of the recorded session time giving cognitive support (e.g., instructions). The major implication for the CBR system design was that because level of support widely varied, there is a need to describe level of expected patient scaffolding (cognitive and physical help) based on game type and patient abilities; this feedback should be included as part of the recommendation results.

c) *Assistive devices*: In most sessions that we observed, patients did not require any assistive devices; however, several people used walkers (17%) and canes (8%). Six patients (25%) were in wheelchairs but were able to stand using a walker to play the games. The major implication for the CBR system design was a need to filter games based on whether they could be played using assistive standing devices and whether the games can be played sitting down.

d) *Timing*: Setting up a game can take a lot of time, and it really varied from game to game. Therapists spent an average of 24% of the session time starting and setting up the games in the observations we coded; time spent on set-up varied between 7% and 38% of session time. The major implication for the CBR system design was a need to include expected start-up times for each game.

C. Step three: Diary study

From the interviews and observations, the concept of a ‘case’ evolved. Findings from the diary studies have clarified the requirements by helping us define the attributes in a ‘case’ and have provided ‘seed cases’ that determine how well (from therapists’ perspectives) the motion-games matched the session goals and patient factors. We are exploring relationships among the following categories of attributes: (1) patient variables, (2) therapy session goals, (3) game/console affordances, mechanics, and requirements, and (4) subjective measures of session outcome (e.g. effectiveness on therapeutic goals), see Fig. 2 for how we are envisioning a case.

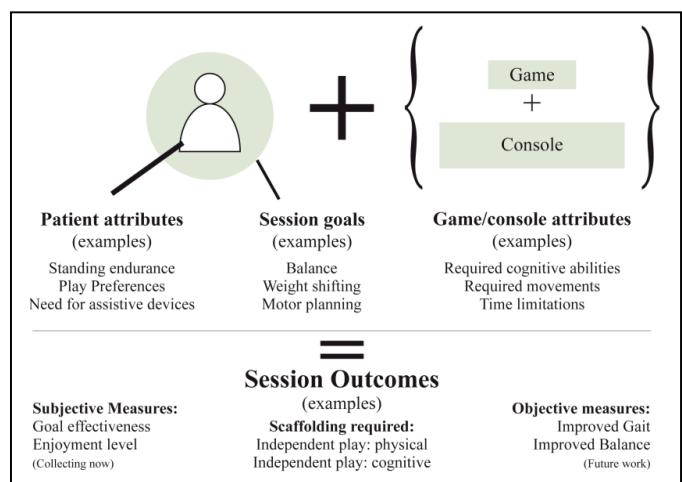


Fig 2. Summary attributes of a ‘case’

Fig 3. Current versions of the diary forms

The diary forms were based on how we defined a case. Therapists were given a notebook containing a two-page paper diary form and were asked to record details about play sessions over two-week periods. The first diary form designs were based on what we learned in the interviews and observations. The diary method has allowed non-identifiable patient data collection across a much wider range of patients than we are permitted to observe. We piloted the diary forms in October 8-19, 2012 at Schwab.

We continued to refine the diary forms (see Fig. 3 for the latest version) through the early studies at Schwab based on therapists' feedback. We collected game session information for three periods at Schwab (December 3-14, 2012, February 18-March 1, 2013 and June 12-29, 2013), and one period to date at Marianjoy (November 4-15, 2013). (As of this writing, three more diary collection periods are scheduled at Marianjoy in March, May and July 2014.)

1) *Participants:* Therapists have recorded data for 77 individual patients, ages ranging from 19-89 ($M = 54.0$); a little less than half ($N = 34$, 44%) were male. Through four studies, we have collected 184 cases; a case is a combination of (1) a patient, (2) a game/mini-game played on a console, and (3) its outcomes. The three most common causes of BI were: (1) stroke ($N = 48$, 62.5%), (2) TBIs ($N = 10$, 13%) due

to car accidents, falls, gunshot wounds and other violently related causes, and (3) disease-related ($N = 10, 13\%$) including encephalopathy and meningitis. Fifteen therapists who had participated in the interviews have recorded diary data. Broken down by occupation: eight physical therapists,, three occupational therapists, two speech-language pathologists, and two recreational therapists.

2) *Procedures:* On the first diary page, therapists were asked to record (1) date/time of the session; (2) session details (e.g., if a group session, the number of patients involved); (3) non-identifiable patient details (e.g., initials, age, gender, balance and gait measures, nature of the BI, assistive devices used, standing endurance, range of motion, fine motor control, command following and sequencing abilities; and (4) the game selections (console, game/mini-game). We also assessed patients' play preferences using work from Stuart Brown [30].

Brown has studied play personalities by conducting 'play histories' which are interviews focused on play [30]. From these studies he distilled eight play personality types (although most people are a combination): (1) Joker whose sense of play involves nonsense; (2) Kinesthete who enjoys movement; (3) Explorer who enjoys new experiences; (4) Competitor; (5) Director who enjoys organizing and planning events; (6) Collector; (7) Artist/Creator; and (8) Storyteller. To explore

patients' play preferences, we created eight cards describing Brown's play personalities in a single sentence. We described each card (after shuffling), laid them out for the player-patients and asked them to identify the one that described them best. We also asked for their second and least identifiers.

On the second diary page, therapists: (1) identified the goals (check-mark) for playing each game (listed goals were based on the interviews, observations and the pilot diary study); (2) rated (subjective measure) the effectiveness (-2 to +2) of the game at meeting specified session goals (we set a rating of 0 as 'effective' after we found that therapists were likely to find the games at least somewhat effective because they chose them, i.e., confirmation bias); (3) rated the level (0-4) of cognitive and physical help needed to play the game; (4) rated patient enjoyment (0-4); (5) assessed appropriate challenge (range from boredom to frustration); and (6) were invited to enter additional comments about the session.

3) Summary Findings and Proof of Concept for CBR system: As a proof of concept, in this summary, we compared two games in which we had the most recorded cases, bowling ($N = 47$, 34 on the Wii, 13 on the Kinect) and soccer ($N = 13$, 1 on the Wii and 12 on Kinect). We are combining results from two soccer mini-games available in Kinect sports: target kick in which players kick into a goal and super saver in which players are the goalie.

TABLE I: SHARED GOALS: BOWLING V. SOCCER
Bowling ($N = 47$) Soccer ($N = 13$)

| | | |
|-----------------|---------------|---------------|
| Weight shifting | 100% of cases | 100% of cases |
| Static balance | 96% of cases | 92% of cases |
| Standing | 72% of cases | 85% of cases |
| Endurance | 70% of cases | 85% of cases |
| Dynamic balance | 74% of cases | 92% of cases |

Therapists had many common shared goals when using bowling and soccer; see Table 1 for goals that were selected for more than 50% of the cases.

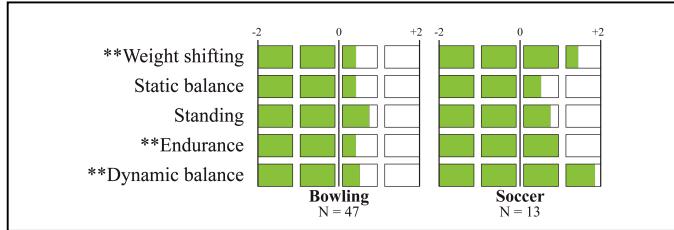


Fig 4. Bowling V. Soccer comparison by common goals

Using Mann-Whitney U tests we compared reported success for each common goal. There was a significant difference in effectiveness of the games to address three common goals: (1) 'weight shifting', $U(60) = 133.0$, $Z = -3.99$, $p < .001$, specifically, soccer was rated higher (average score on a scale of -2 to +2 = 1.23) than bowling (average score was - .23); (2) 'endurance', $U(57) = 103.0$, $Z = -2.24$, $p < .05$, specifically, soccer was rated higher ($M = 1.00$) than bowling ($M = .74$); and (3) 'dynamic balance' $U(35) = 63.0$, $Z = -3.81$, $p < .00$, specifically, soccer was rated higher ($M = 1.92$ – this

was soccer's highest rated goal) than bowling ($M = .69$). No other common goals were rated as significantly different; however, the most successful reported goal (among goals with more than 10 reports) for bowling was for socialization ($M = 1.06$). See Fig. 4 for comparisons by common goals.

To explore the personality matching aspect of the seed cases, we examined differences among player types (as defined by the first choice card) and level of enjoyment when considering all gaming, the differences were significant, $\chi^2(1, N = 35) = 58.75$, $p < .05$. Specifically, Jokers ($N = 63$) were rated as enjoying gaming more than other play personality types, (of a score from 0-4, $M = 3.19$). Kinesthetes ($N = 19$) enjoyed gaming overall the least ($M = .92$).

Together, these findings indicated that if therapists wanted to improve dynamic balance and had a patient who identified as a Joker-type, a soccer game would be a better choice than bowling. However, if socialization were also important, bowling might be a better choice. Similarly, if game developers wanted to design a game to improve dynamic balance for Joker-typed patients, they could benefit from game design elements from soccer; if they want to focus on socialization, game elements in bowling would be more beneficial. In other words, game developers can utilize the same data to gather personalized requirements of motion-games for BI therapy.

D. Step four: User Studies of a Prototype CBR User Interface

User studies of the system interface have helped us to understand interface and content requirements and further clarify the attributes of a case. We designed the initial CBR user interface based on the diary form and have evolved the interface based on therapists' feedback. We first evaluated paper-based wireframes (line drawings) of the interface with three therapists onsite at Schwab (one each occupational, recreational and physical therapist). Based on the feedback we created a responsive interactive web-based wireframe to usability test the interaction.

The interface's introduction page indicates that the system provides three major functions.

1. *Find game recommendations*, in which the system recommends suitable games for a user based on patient attributes and therapeutic goals.
2. *Explore games*, in which users can use filtering, and search tools to find information about games.
3. *Contribute to the system*, in which users can add information about a game-play session, add a comment to a game, or add a new game.

The process of finding game recommendations for patients involved two steps, which resulted in a summary recommendation. The steps were: (1) provide session information including physical and cognitive goals, and determine group size; and (2) provide patient information (that could be skipped) including initials, age, gender, balance and gait measures, nature of the BI, assistive devices used, standing endurance, range of motion, fine motor control, command following and sequencing abilities, and their play personalities using the Brown method (described above). Once the summary recommendation was displayed, more information was

Fig 5. Selected Screenshots of Prototype UI (available at http://cyputnam.com/CBR_UI/)

available on demand, see Fig. 5 for selected screen captures of the interactive wireframes (wireframes also available at http://cyputnam.com/CBR_UI/)

1) *Procedures*: We conducted usability studies with four therapists who were involved in the original interviews (one speech-language pathologist, one recreational and two physical therapists). Usability studies were conducted onsite at Schwab. Participants were asked to complete three tasks using a think-aloud protocol [31]:

1. Think about a patient you worked with recently, who could potentially benefit from playing games. Use the system to choose two games that will work well for your patient.
2. You heard about a game called 'Space Pop' on Xbox 360 Kinect and think it might be appropriate for some of your patients. Use this system to answer the following questions about this game:
 - a. Is Space Pop playable by a patient who uses a wheelchair?

b. Is Space Pop recommended for improving fine-motor abilities?

c. How long will you need to spend to set up Space Pop? How long will it take to play Space Pop?

3. Use the system to find games that satisfies the following three conditions:

- a. Possible to play with two players (your patient and you)
- b. Playable by a patient who uses a wheelchair
- c. Takes about three minutes to play.

After the usability studies, therapists were asked: (1) to give us open-ended feedback about the system; (2) which features they found most useful; (3) features they felt were missing; and (4) if they had other questions and comments.

2) *Summary findings*: All participants were able to complete the three tasks, and were generally positive about the concept. For example participant 3 told us, "**For being not very strong in this world, I love the idea of it – that I don't have to know**

stuff, that I don't have to have a background in any of it and I can still pull out what I could potentially use for a various of patients. I love that you can either be specific or not.”

The most useful feature mentioned (beyond the recommendations themselves) was the inclusion of the expected cognitive help; however, one therapist (participant 4) wanted us to take this further and include sensory attribute of patients, *“Maybe it would be helpful to add some kind of sensory integration element, because we do have patients who are both hearing and visually impaired.”*

Missing features included: (1) a way to filter out violent games, (2) better filtering by game genre, and (3) a desire to reorder the feedback so that the ‘goal effectiveness’ (how well the game was judged to effectively target a goal) and the level of expected independent play (both cognitive and physical) were at the top of the results page.

E. Summary of the four-step approach

With an emphasis on direct involvement of end users (i.e. therapists), each step in our multi-method approach has helped us understand different aspects of requirements for the CBR system. See Table 2 for a summary of the steps and the elements of requirements they afforded.

TABLE II: SUMMARY OF REQUIREMENT GATHERING STEPS

| Steps | Elements of Requirements |
|------------------------|---|
| Interviews | The CBR system needs to consider therapy session goals, patient physical abilities and cognitive abilities |
| Observations | The CBR system needs to incorporate patient scaffolding aspects and logistic issues in therapy |
| Diary Studies | Clarified the attributes in a ‘case’ (e.g. specific patient abilities needed to be considered) and generated ‘seed cases’ |
| Usability Tests | Clarified the user interface content requirements and elaborated the attributes in a case |

Described here, we have presented four research steps as though they proceeded linearly. However, in reality the process has been very iterative. The interviews we conducted at Marianjoy were similar to those at Schwab, but were changed with what we had learned at Schwab. The observation coding method we used for the Marianjoy videos was first piloted with the Schwab videos. The first (pilot) diary forms we used at Schwab were comprised primarily of open-ended questions. Consequently the pilot diary data was only used to inform later versions of the diary form; i.e., we modified the design to help minimize the time it took for therapists to complete the form. We continued to iterate the form throughout the first three diary studies at Schwab based on therapists’ feedback.

There was also great overlap and interaction between the method steps. For example, both step three (diary studies) and step four (interface prototype studies) contributed to the evolution of both the diary form and the interface design. We designed the first version of the interface based on the iterated

diary form at Schwab. After the user studies at Schwab, we then turned around and made further modifications to the diary form that was presented in this paper to make it look and function more like the interface. We feel this interaction between requirements gathering and user studies was crucial for our project.

IV. DISCUSSION AND CONCLUSIONS

We recognize that choosing methods for requirement elicitation is dependent on characteristics of a project domain. While some researchers have proposed frameworks that can be used for choosing elicitation methods (e.g. [15]); others (similar to this work) have reported their success in combining elicitation methods in various domains [13, 32]. For example, Sabahat et al. analyzed the strength and weakness of elicitation techniques for gathering different requirement aspects in the domain of Global Software Development [32]. While focusing on a different domain, we highlighted similar problems of requirements gathering for a special population.

We initially chose more exploratory methods (i.e. interviews and observations) because of our limited knowledge and lack of context on the problem domain (i.e. therapeutic gaming in BI rehabilitation) at the outset of this project. As the project proceeded and shifted focus towards a potential solution (CBR system), we adapted and refined the interviews and observations to focus on elicitation of the requirements for the CBR system. In other words, the lens for the data collection shifted from problem to solution exploration. However, we found we needed a different type of method to better understand the patients’ attributes in the system because it is not possible for us as researchers to directly interact with many patients; i.e., those who are not capable of signing consent forms. The diaries then became a proxy means to gather patient attributes for the CBR system. The diaries were not just helpful to leverage therapists’ expertise, but the diary design evolution, which was based on therapist feedback, directly led to the initial designs of the CBR prototype interface.

While it is obvious that direct interaction with end users is a critical key to success of technologies, it is even more important when considering requirements elicitation for projects that involve populations that are less represented in software engineering studies. In our case, the end users are the therapists who use games with their patients who have had BIs. In each step of our multi-method approach, therapists have been and will be deeply involved; their opinions and domain knowledge is fully acknowledged:

- In the interviews, user requirements that consider therapeutic goals and patient factors (e.g. patient abilities and personal preferences) are directly derived from therapists’ responses.
- In the observations, both therapists and patients are involved. We gathered details about the interaction between therapists and patients in situ (e.g. physical scaffolding).
- In the diary studies, the diary form itself (which is later used as a framework for the interface design) has evolved based on therapists’ feedback.

- In the user study of the interface prototype, usability issues and missing elements have been identified with the support of users.

Moving forward, this project also will consider the needs and domain expertise of game developers. There are two major approaches in the domain of therapy-centered motion gaming: (1) using existing commercial games with no modifications; and (2) creating new games specifically for rehabilitation [9, 33-35]. Both approaches have merit. While acknowledging the limitations of commercial motion-games (e.g. learning curve is too steep and sometimes physically and/or cognitively too challenging [36,37]), we recognize that commercial games, when carefully chosen for patient characteristic and therapeutic goals, can benefit a very wide range of people with BIs because they are readily available. As such, in the current work we are focusing on the first approach (i.e. direct use of commercially available motion-games for BI rehabilitation) and aim to support therapists in choosing commercial games.

However, we envision that our work will also benefit the second approach (i.e. creating games for BI rehabilitation). In particular, our work supports a personalized requirement gathering of therapy-centered motion games that will aid game developers interested in designing therapy-centered games. As our proof of concept analysis of the diary data has demonstrated, play personalities can significantly influence gaming preference and gaming experience. In addition, patients' physical and cognitive abilities affect the accessibility of a game and different therapeutic goals lead to different levels of effectiveness in therapy outcomes. Game developers need to consider all these factors in order to produce therapy-centered games that are both (1) enjoyable for this diverse population and (2) effective for the aimed therapeutic goals. We view the CBR system as part of the ecology of approaches to therapy-centered gaming.

V. LIMITATIONS AND FUTURE WORK

As with all research, this work has several limitations; some we will continue to address moving forward. Limitations of this work include: (a) a specialized domain; (b) small sample size; (c) reliance on subjective measures and (d) as a work in progress, this project is still evolving.

A. Specialized Domain

In this paper, we focused on a case study addressing requirements elicitation problems of one project that leverages user-centered approaches and concentrates on a specialized domain. While our approach might be beneficial to similar projects that focus on requirements gathering for an unfamiliar and less represented population, the external validity is limited.

B. Sample size

Our sample size for most games (other than bowling and soccer) was too small to infer how well the games met any particular goal; i.e. there was no power to see statistical differences in many cases. Additionally, the requirements we gathered from Schwab and Marianjoy may not generalize to other rehabilitation hospitals or to other facilities (for example, outpatient clinics). We plan on addressing this limitation by

expanding to additional rehabilitation hospitals and including outpatient care.

C. Reliance on subjective measures

Diary study data relied completely on subjective measures; i.e., on therapists' opinions about game success and patient enjoyment. While subjective opinions are very important (and in fact drive most recommendation systems), in future work objective measures of game efficacy are needed to credibly discuss motion-game efficacy. We are currently partnering with biomedical scientists at Rosalind Franklin University to pilot the collection of gait and balance measures during the diary studies at Marianjoy. The addition of objective measures through experimental design in outpatient care is also part of our future plans.

D. Work in progress

We acknowledge that because this is work in progress, our approach and definition of 'a case' will probably need to evolve given the diversity of brain injuries, patient characteristics, and game features. This will be addressed through evaluation studies and iterative design of the CBR system with users (therapists and game developers). We plan to launch an early beta of the system (revised based on therapists' feedback) in winter 2015.

Additionally, to further support personalized requirement gathering, we plan to create an interface to the CBR database specifically for game developers. The game designer interface will afford an understanding of common game design requirements (e.g., common goals, effective movements) for therapy-centered motion games.

ACKNOWLEDGMENT

This work was funded by the DePaul University Research Counsel and by a joint grant from Rosalind Franklin and DePaul Universities. We would like to thank all of our therapists and patient participants and our partners in the Quality and Research offices at the hospitals. Thanks also to the reviewers whose careful input helped to improve this paper.

REFERENCES

- [1] Centers for Disease Control. "Traumatic Brain Injury in the United States: Fact Sheet," available: http://www.cdc.gov/traumaticbraininjury/get_the_facts.html, last accessed March 9, 2014.
- [2] Brain Injury Association of America. "Quick Facts About Brain Injury", available: <http://www.biausa.org/bia-media-center.htm>, last accessed March 9, 2014.
- [3] J. M. Silver, T. W. McAllister, and S.C., Yudofsky, S. C., Textbook of Traumatic Brain Injury, 2nd ed. Arlington, VA, USA: American Psychiatric Publishing, Inc, 2011.
- [4] Brain Injury Association of America. "Living With a Brain Injury", available: <http://www.biausa.org/living-with-brain-injury.htm>, last accessed March 9, 2014.
- [5] M. Selzer, S. Clarke, L. Cohen, P. Duncan, and F. Gage, Eds., "Textbook of Neural Repair and Rehabilitation," Cambridge University Press, New York, NY, USA, 2006.

- [6] J.-A Gil-Gomez, J.-A. Lozano, M. Alcaniz, and S.A. Perez, "Nintendo Wii Balance board for balance disorders," presented at the Virtual Rehabilitation International Conference, Haifa, 2009.
- [7] J.-A Gil-Gomez, J.-A. Lozano, M. Alcaniz, and C. Colomer, "Effectiveness of a Wii balance board-based system (eBaViR) for balance rehabilitation: a pilot randomized clinical trial in patients with acquired brain injury," *J Neuroeng Rehabil*, vol. 8, May 23 2011.
- [8] A. L Betker, A. Desai, C. Nett, N. Kapadia, and T. Szturm, "Game-based Exercises for Dynamic Short-Sitting Balance Rehabilitation of People with Chronic Spinal Cord and Traumatic Brain Injuries," *Physical Therapy*, vol. 87, pp. 1389-1398, 2007.
- [9] G. Alankus, A. Lazar, M. May, and C. Kelleher, "Towards Customizable Games for Stroke Rehabilitation," presented at the Conference on Human Factors in Computing Systems, CHI '10, Atlanta, GA, USA, 2010.
- [10] S. Flynn, P. Palma, and A. Bender, "Feasibility of Using the Sony PlayStation 2 Gaming Platform for an Individual Poststroke: A Case Report," *Journal of Neurological Physical Therapy (JNPT)*, vol. 31, pp. 180-189, 2007.
- [11] G. Alankus, R. Proffitt, C. Kelleher, and J. Engsberg, "Stroke therapy through motion-based games: A case study," *ACM Transactions on Accessible Computing*, vol. 4, November, 2011 2011.
- [12] J.A. Goguen and C. Linde, "Techniques for Requirements Elicitation" Requirements Engineering, 1993., Proceedings of IEEE International Symposium, pp.152,164, 4-6 Jan 1993.
- [13] D. Mishra, A. Mishra, A and A. Yazici., "Successful requirement elicitation by combining requirement engineering techniques," *Applications of Digital Information and Web Technologies, ICADIWT 2008*, pp.258,263, 4-6 Aug. 2008.
- [14] A. Davis, O. Dieste, A. Hickey, A., N. Juristo, A.M. Moreno, "Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review," *Requirements Engineering*, 14th IEEE International Conference, pp.179,188, 11-15 Sept. 2006.
- [15] S. Tiwari, S.S. Rathore, A. Gupta, "Selecting requirement elicitation techniques for software projects," *Software Engineering (CONSEG)*, 2012 CSI Sixth International Conference, pp.1,10, 5-7 Sept. 2012.
- [16] N. Seyff, F. Graf, N. Maiden, "Using Mobile RE Tools to Give End-Users Their Own Voice," Requirements Engineering Conference (RE), 2010 18th IEEE International, pp.37,46, Sept. 27 2010-Oct. 1 2010.
- [17] R. Alvarez, "Discourse analysis of requirements and knowledge elicitation interviews," *System Sciences*, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference, pp.10 pp., 7-10 Jan. 2002.
- [18] O. Brill, and E. Knauss, "Structured and unobtrusive observation of anonymous users and their context for requirements elicitation," Requirements Engineering Conference (RE), 2011 19th IEEE International, pp.175,184, Aug. 29 2011-Sept. 2 2011.
- [19] L. Silva, M. Borges, and P. de Carvalho, "Collaborative ethnography: An approach to the elicitation of cognitive requirements of teams," *Computer Supported Cooperative Work in Design*, 2009. CSCWD 2009. 13th International Conference, pp.167,172, 22-24 April 2009
- [20] S. Shahidi, and Z. Mohd Kasirun, "Using Ethnography Techniques in Developing a Mobile Tool for Requirements Elicitation," *Information Management and Engineering*, 2009. ICIME '09. International Conference, pp.510,513, 3-5 April 2009
- [21] Fickas, S., "Clinical requirements engineering," *Software Engineering*, 2005. ICSE 2005. Proceedings. 27th International Conference, pp.28,34, 15-21 May 2005.
- [22] A. Sutcliffe, S. Fickas, and M.M. Sohlberg, "Personal and contextual requirements engineering," *Requirements Engineering*, 2005. Proceedings. 13th IEEE International Conference, pp.19,28, 29 Aug.-2 Sept. 2005.
- [23] I. Watson, "Applying Case-Based Reasoning: Techniques for Enterprise Systems", San Francisco, CA, USA: Morgan Kaufmann Publishers, 1997.
- [24] A. Holt, I. Bichindaritz, R. Schmidt, and P. Perner, "Medical applications in case-based reasoning," *The Knowledge Engineering Review*, vol. 20, pp. 289-292, 2005.
- [25] M.U. Ahmed, and P. Funk, "A computer Aided System for Post-operative Pain Treatment Combining Knowledge Discovery and Case-Based Reasoning," in *ICCBR*, Lyon, France, 2012.
- [26] J. Saldaña, "The coding manual for qualitative researchers". Thousand Oaks, CA, USA: Sage, 2009.
- [27] D. G. Altman, "Practical statistics for medical research", London: Chapman and Hall, 1991.
- [28] D.J.Walsh, N. Bakin, T.B. Lee, Y. Chung, and K. Chung, "Using digital video in field-based research with children: A primer", In *Early childhood qualitative research* (pp. 43-62), 2006.
- [29] B. S. Gardner, "Responsive Web Design: Enriching the User Experience." *Connectivity and the User Experience*, 13, 2011.
- [30] S. Brown, "Play: How it shapes the brain, opens the imagination and invigorates the soul", New York, NY, USA: Penguin Group, 2009.
- [31] J. Nielsen. "Think Aloud: The #1 Usability Tool", available <http://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/>, last accessed March 9, 2014
- [32] N. Sabahat, F. Iqbal, F. Azam, F, and M.Y., Javed, "An iterative approach for global requirements elicitation: A case study analysis," *Electronics and Information Engineering (ICEIE), 2010 International Conference*, pp.V1-361,V1-366, 1-3 Aug. 2010
- [33] Jintronix, available <http://www.jintronix.com/>, last accesses March 9, 2014
- [34] J.W. Burke, M.D. McNeill, D. K. Charles, P.J. Morrow, H. Crosbie, and S. M. McDonough, "Serious Games for Upper Limb Rehabilitation Following a Stroke," in *Games and Virtual Worlds for Serious Applications, VS-GAMES '09*, Coventry, UK, pp. 103 -110, 2009
- [35] B. T. Horowitz, "Blue Marble Games Aid People With Traumatic Brain Injuries", *dWeek*. Available: <http://www.eweek.com/enterprise-apps/slideshows/blue-marble-games-aid-people-with-traumatic-brain-injuries>, 2013.
- [36] J.W. Burke, M.D. McNeill, D. K. Charles, P.J. Morrow, H. Crosbie, and S. M. McDonough, "Optimising engagement for stroke rehabilitation using serious games," *The Visual Computer*, vol. 25, pp. 1085-1099, 2009.
- [37] D. Rand, R. Kizony, and P.L. Weiss, "Virtual reality rehabilitation for all: Vivid GX versus Sony PlayStation II EyeToy," in *Proc. 5th Intl Conf. Disability, Virtual Reality & Associated Tech*, Oxford, UK, pp. 87-94, 2004.

Towards a Situation Awareness Design to Improve Visually Impaired Orientation in Unfamiliar Buildings: Requirements Elicitation Study

Abdulrhman Alkhanifer

Computing and Information Science Program
Rochester Institute of Technology (RIT)
Rochester, New York 14623, USA.
alknaifer@mail.rit.edu

Stephanie Ludi

Software Engineering Department
Rochester Institute of Technology (RIT)
Rochester, New York 14623, USA.
salvse@rit.edu

Abstract—Requirements elicitation can be a challenging process in many systems. This challenge can be greater with a non-standard user population, such as visually impaired users. In this work, we report our experience and results of eliciting user requirements for a situation awareness indoor orientation system dedicated to the visually impaired. We elicited our initial system requirements through three different studies that focus on users along with orientation and mobility instructors. Also, we performed a knowledge elicitation through our studies to formulate our system's situation awareness requirements.

Index Terms—Visual impairment, requirements elicitation, situation awareness requirements, assistive technology, and qualitative analysis.

I. INTRODUCTION

The World Health Organization (WHO) estimated that there are currently 285 million people who have a visual impairment [4]. Visually impaired (VI) persons can be either legally or totally blind. Legally blind means having the visual acuity of 20/200 in the better eye due to a vision problem that cannot be corrected, a limited vision field, or a visual disorder [3]. While sighted people rely on their vision to understand and orient themselves in unfamiliar indoor environments, VI persons rely on their other senses to understand such environments. This reliance, however, causes many challenges that this class of users faces. For instance, VI users might get distracted and veer from their path in any unfamiliar, noisy indoor environment. VI challenges in the unfamiliar buildings motivate us to develop an assistive orientation technology that helps to raise their environmental situation awareness of unfamiliar indoor environments, and therefore accommodate user's orientation. Situation awareness (SA) is known as the “the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future” [16]. Due to the scope of this paper, we will not discuss the SA design.

Orientation is “the ability to use one's remaining senses to understand one's location in the environment at any given time [1].” Visually impaired users face many orientation challenges when they enter unfamiliar indoor environments. Tasks, like

for example, realizing the surrounding important landmarks, can be difficult in unknown environments as users try to pick up cues and comprehend them to build their mental model. A mental model is a cognitive term that refers to the phenomenon of having an imaginary mental image that depends on a person's knowledge and experiences [7].

The SA design approach was selected for this work for two reasons. First, the SA design approach is useful for systems that support user's decisions. Second, SA design helps in deciding what and when to present information to the users, while maintaining an acceptable information flow that allows the users to comprehend presented information [5]. The VI need to be able to focus on the environment while using our system.

Requirements elicitation with a non-standard population poses many challenges as software engineers need to understand the problem domain. With VI users, many challenges are presented such as the absence of visual aids to discuss user's needs. In this study, we discussed our work with two different stakeholder populations to elicit and define initial requirements for an SA orientation assistive technology to enhance VI orientation in unfamiliar indoor environments. The system goal is to enhance VI user's performance in unfamiliar indoor environments by providing them with an assistive technology that helps in raising their SA of the surrounding environment. To elicit the system requirements we performed a series of three requirement elicitation studies: domain understanding, orientation and mobility (O&M) recommendations, and survey-based studies. O&M is defined as teaching VI persons the required skills to be able to travel safely and efficiently in any environment [2]. Eliciting O&M instructors' knowledge is very important for our design as they can provide recommended guidelines that help in enhancing VI user's mobility and safety when orienting in unfamiliar indoor environments. Our elicitation studies' goal is to realize user's problems and to elicit initial user and SA requirements.

Our contribution in this paper can be summarized as conduction of the SA requirements' elicitation and analysis of VI users' orientation in unfamiliar indoor environments. Our results can help other researchers who are conducting further

SA applications in the field. The rest of this paper will discuss in detail our elicitation process and results of this research. Section two will provide a literature review of the field of requirements elicitation for VI indoor orientation. Section three will illustrate our methodology that captured and elicited the initial requirements. Sections four, five, and six will discuss our elicitation study phases in details. Section seven will briefly outline some user's needs diversity based on the geographical location. Section eight will discuss users profile. Section nine will overview our elicited SA requirements, and section ten will discuss lessons learned from our work.

II. LITERATURE REVIEW

Previous research has been conducted to elicit VI user's requirements for assistive technologies to raise the user understanding or guide him/her through different indoor and outdoor environments. However, to the best of our knowledge, none of the previous work has been addressed to elicit SA requirements for VI indoor orientation. SA requirements help when designing systems that support user's decisions and actions, such as orientation assistive technologies dedicated to the VI. We also performed semi-structured interviews with six O&M instructors, which provided us with critical insight to help design our system. The rest of this section will discuss some of the previous work in requirements elicitation with VI users for the purpose of providing an aid to the users when orienting in indoor environments. Also, we will discuss one related work in the field of goals and requirements analysis for assistive technologies.

Miao, et al. [12] reviewed their conducted requirements for an indoor navigation system dedicated to the blind users. Their motivation was to enable blind users to be independent travelers by providing them with an indoor navigation system. To decide their user profile, they surveyed six blind users. They also interviewed one mobility instructor and investigated two environments where blind users travel. For their users' requirements, they conducted structured interviews with six blind users. Through the interviews, they investigated what functionality could be offered by their system and how to present the system information. Authors elicited user requirements through structured interviews, which might restrict users to elaborate on related issues to the questions being asked. Also, the authors interviewed only one mobility instructor. In our work, we used semi-structured interviews where users elaborated upon our interview questions, which generated some important ideas. Also, we interviewed six O&M instructors and got their insights on the recommended ways to tackle the orientation problem in unfamiliar environments.

Rafael, et al. [11] explained their development of a prototype to enable VI users to identify people around them using a mobile-based Bluetooth. Before conducting their prototype, they elicited user requirements through interviews with 19 blind users. Interviews were focused to realize user problems in identifying objects and people around in both indoor and outdoor environments. In their elicitation findings, they reported user needs towards identification of objects and

people in the environment. The authors' work was directed to assist blind users to dynamically identify other people in the surrounding environment. This, however, is different than our focus, which is on raising VI user's SA in unfamiliar indoor environments by enabling them to understand the environment dynamically as they travel in it.

Strohotte, et al. [8] and Johnson and Petrie [9] discussed the elicitation study for the MoBIC (Mobility of Blind and Elderly People Interacting with Computers) project. They performed their elicitation with the help of 24 VI users, and instructors. Some of the VI users were cane users, while the rest were guide-dog users. Their study's goal was to identify the habits and problems that exist with VI users traveling in familiar and unfamiliar environments. The authors used interviews to elicit user requirements, which consisted of open and close-ended questions. Their study shows a high VI demand for an assistance detecting indoor and outdoor obstacles, and for help crossing streets. While this work focuses on the safety travel of VI users in any environment, it does not discuss orientation and obstacles in the unfamiliar indoor environments. In our work, we focused on the VI orientation in unfamiliar indoor buildings as users behave differently as opposed to known environments.

O'Neill, et al. [10] and Engelbrektsson, et al. [14] discussed the Personal Adaptive Mobility Aid (PAM-AID) project, which is intended to support indoor mobility and navigation of elderly and frail VI users. Their goal was to make users independent of caregivers by providing users with the physical support as well as guidance. They performed an initial requirement elicitation study in three countries: Ireland, Sweden, and the UK. The authors performed face-to-face interviews with 38 users who live in 14 care facilities. They also interviewed 14 professionals who work in the care and rehabilitation centers as their secondary stakeholders. Their interviews focused on two aspects. First were the current available assistive technologies and aids. Second was user's preference for his/her system's input/output interactions. This work is dedicated to serve a special group of VI users: elderly and frail. Elderly and frail VI users can navigate unfamiliar indoors with the help of care persons, or using an equipped wheelchair. In our work, we focus on a different class of users: VI cane users who are able to walk unassisted and not hard of hearing or deaf. Also, we performed a domain analysis using O&M instructors' interviews, as they provided us with how to tackle orientation problems in unfamiliar indoor.

Sutcliffe, et al. [17] proposed a three-layer requirement's analysis framework that focuses on user's personal goals as well as how a system achieves user's goals. The first layer focuses on the general goals of stakeholders. The second layer focuses on user characteristics such as physical context details. The third layer focuses on specific personal goals that vary from one user to another. The authors performed two assistive technology case studies where they used their proposed framework to perform the requirements analysis: email and a navigational support application. Their navigational support application was addressed to support users who suffer from traumatic brain injuries, which sometimes result in short-term

memory loss. In this case study, authors identified requirements under three levels. The first level focuses on the general goal of users who need assistance when traveling to public transit modes. The second focuses on user characteristics such as the memory loss. The third focuses on users' personal goals such as travel purpose. The framework provided by the authors is beneficial for systems that need to be customized on a personal level to the needs of each user. In our work, we identified users' goals using SA analysis and then identified decisions and information that assist each goal. SA design provides a good method to support user decisions.

III. METHODOLOGY

Our methodology relies first on qualitative analysis that uses the content analysis [13], and second on an open-coding technique, which is a part of the grounded theory method [1]. We used an iterative fashion to re-evaluate and re-define our research questions and user's problems. In the rest of this section we will discuss briefly our elicitation process and SA requirement analysis methodology. To minimize the bias in the elicitation process, we designed our questions in a way that does not lead the participants to specific answers. Also, each interview results (phase 1 and 2) were validated and expanded in the following phase.

A. Requirements Elicitation

We used an ethnographic interview method [18] as well as surveys to elicit system's requirements. Two of our elicitation studies were in the form of semi-structured phone interviews, and the other one was an online survey. Our studies focused on users as our primary stakeholders and O&M instructors as our secondary stakeholders. Eliciting requirements from different stakeholders can bring diversity to our system's requirements, as users provide their needs and O&M instructors provide us with the domain efficient and safe practices in indoor orientation.

Every elicitation phase's results were fed into the next one to tailor the questions towards elaboration on user's reported requirements. The questions were designed not to lead users to any preferences; instead, they explore the problems from the stakeholders' point-of-view. All questions that were presented to the users during the course of elicitation corresponded to our research questions and goals. Our study was designed to investigate many aspects that relate to orientation in unfamiliar indoor environments, such as: user's orientation issues, past experience with indoor navigation systems, experience with obstacles detecting technologies, familiarity and experience with smartphones, and user's preferences towards system interaction.

To analyze open-ended questions responses, we analyzed their content and looked for similarities. Then, using the open-coding method [1] we developed a set of codes that represent ideas or problems that were reported through our interviews. Afterwards, we started assigning codes to each participant's answers. Later, codes were calculated depending on the number of their occurrences. Higher occurrences indicate higher importance or demand.

With participants' permissions, we recorded our interviews while maintaining participants' anonymity. Each interview took between 40-60 minutes. Participants were recruited randomly through online mailing lists dedicated to the VI users. The recruiting process for each phase was performed independently.

B. Situation Awareness Requirements

The approach we use for our SA requirement analysis is based on the method described by Endsley and Jones [5]. They recommend the use of Goal-Directed Task Analysis (GDTA) as a method to elicit user's knowledge. This knowledge includes user goals, decisions, and information that is needed to support each goal and sub-goals. Crandall, et al. [7] describes GDTA method as one of the ways to conduct Cognitive Task Analysis (CTA) that allows the elicitation of user's knowledge. In our study, while eliciting system requirements, we focused on user goals when entering unfamiliar indoor environments. Goals were divided into sub-goals that make it easier to identify the required information that supports them. In our approach, we defined user's goals and sub-goals when entering unfamiliar indoor environments, and associated each sub-goal with the corresponding information needed to achieve it. This analysis was performed in parallel with our system requirements elicitation. While system requirements define the needed requirements to build the system, SA requirements help in knowing what information is needed by users for each goal and how important each piece of information is when reaching each goal.

IV. DOMAIN UNDERSTANDING PHASE

The domain understanding is an important part of our study as it helps researchers to understand the challenges that persist with VI users in unfamiliar indoor environments in different parts of the world. To achieve that, we designed an interview that focuses on the problem and allows us to discuss and elaborate on user problems in such environments. Our designed interview was comprised of 45 close-ended questions and 21 open-ended questions. We interviewed 24 VI across six different countries: US, Canada, UK, Italy, Australia, and New Zealand. Interview questions focused on many topics that relate to the VI orientation in unfamiliar indoors such as: orientation and navigation tasks, users' orientation problems, users' experience with orientation assistive technologies, and system interaction preferences. Interviews were conducted on a one-to-one basis. Each interview took about an hour. There were 10 males and 14 females. The mean age was 49.2 years with a 14.8 years standard deviation.

To understand user needs, participants were asked to discuss and explain scenarios where they navigate an unfamiliar indoor environment. When needed, researchers asked elaboration questions to unveil any unclear points. User scenarios were transformed later in the form of Hierarchical Tasks Analysis (HTA) [7] that models and describes the required steps to perform orientation and mobility tasks in unfamiliar buildings.

Another part that we focused on in our domain understanding study was to measure user acceptance to

different orientation assistive technology forms. This is an important step to our initial requirements, as it will provide us with insight into the user technology preferences for such technology. The rest of this section will explain our results in details.

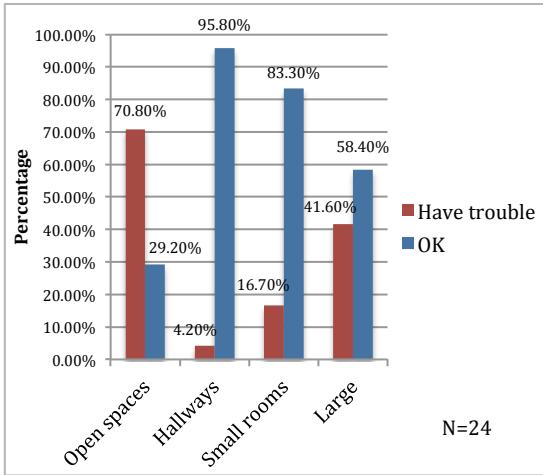


Figure 1. Users expressed their comfort level navigating different unfamiliar indoor setups.

A. Navigating Different Indoor Setups

To focus our design on an indoor setup, we discussed with our interviewed users what indoor setups are most challenging to them. Our results suggest that indoor open spaces represent one of the greatest challenges. Figure 1 illustrates what VI users think about different indoor setups. Absence of landmarks in open spaces was reported as the major reason why open spaces, such as atrium areas, are considered more challenging to navigate through than other indoor setups.

B. User Needs

Users expressed their need for an assistive technology that enables them to navigate unfamiliar indoor environments. In fact, 75% of users who participated in this study stated that they will use any indoor navigation assistive technology, while around 21% said they may use it. Through this study, we captured many user needs when navigating unfamiliar buildings. Primary user needs can be classified as the follows:

- Building information: many users expressed the need for a system to provide them with the unfamiliar building information such as: building layout, number of floors, availability of Americans with Disabilities Act (ADA) bathrooms
- Orientation information: another discussed need was user orientation when the user is in motion. This is vital for users to understand building context as opposed to their position when in motion
- Surrounding important landmarks: users would like to know about the existence and location of important landmarks when the user is in motion. Important landmarks include but are not limited to staircases, elevators, hallways, and bathrooms

- Guidance directions: all users expressed their need of a guiding medium to direct them to their indoor destination. While most of the users preferred one-to-one instructions that are similar to the Global Positioning System (GPS) devices, two users preferred a descriptive guidance that explains user's path and its context
- Obstacles information: users have reported that they need to know more about obstacles in their way. While they can detect many obstacles with their canes or guide-dogs, some obstacles are hard to detect. Examples of these hard-to-find obstacles are obstacles above the user's waist and safety signs such as wet floor signs

C. Assistive Technology Forms

Another area that we focused our elicitation study on was the orientation assistive technology form. Such technologies can be designed in different forms to suit VI users. Examples of the available forms are: wearable, and smartphone-based technologies. User acceptance was measured through interview questions that assess the user's desire to use such technology. System's forms were explained to the participants using different verbal scenarios. Figure 2 shows user preferences towards different orientation assistive technology forms that were presented to the users during the interviews. Users were able to choose more than one preferred technology form.

Another aspect that we looked at was to define our user profiles. At the level of the domain understanding, we were targeting any VI user; however, by looking at our conducted HTA, we discovered different user needs between those who use a guide-dog and those who use a cane. A fundamental difference was that guide-dog users were able to navigate unfamiliar indoor environments in a faster and easier way than cane users. The reason behind that is the ability of guide-dogs to explore and navigate areas. Guide-dog owners can instruct their guide-dogs to search for most of the important landmarks around. For example, a user can instruct his/her guide-dog to find the nearest stairs or elevators. On the other hand, cane users rely on their other senses to understand the environment.

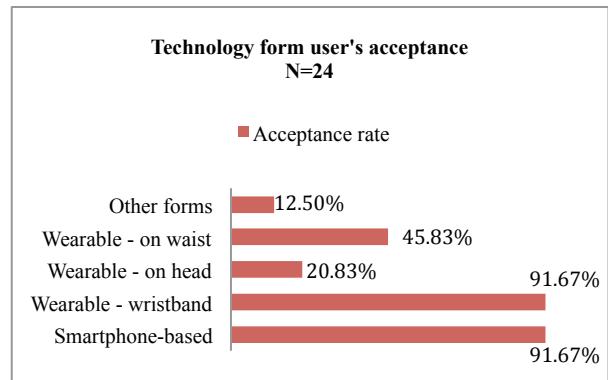


Figure 2. Users acceptance rates to different orientation assistive technology forms that was presented to them through the interviews

Due to the high need for a system from users who use a cane compared to guide-dog users, we decided to dedicate our system to fit this group of users. Another consideration was to consider developing a hands-free assistive device. The reason behind our consideration is that it is extremely dangerous for VI cane users to use anything in their other hand if they were to fall they can protect themselves. This means that they need to interact with a hands-free-device when they are in motion. Also, we considered narrowing our user profiles to any VI users who are able to walk unassisted except hard of hearing and deaf. We excluded hard of hearing and deaf VI users as they need a different way to interact with the system. On the other hand, the ability to walk unassisted was preferred due to the researchers' desire to test the system with users at a later stage.

V. PHASE 2: ORIENTATION & MOBILITY RECOMMENDATIONS

As we started the requirement elicitation by conducting the domain understanding and initial user needs from our system's primary stakeholders, we had to look from a different angle and realize the system requirements from secondary stakeholders' point-of-view. This is an important step, as the system should accommodate VI users with information that supports O&M goals to maintain their safety and to enhance their performance. O&M training is offered by O&M certified instructors who spend months with VI students, teaching them the required skills. Therefore, we designed our second elicitation study to target O&M instructors and elicit system requirements from their point-of-view.

We interviewed six O&M instructors who live in the US. Instructors were distributed around five different geographical locations. We designed interview questions to capture the instructors' point of view of which O&M training techniques are recommended to enhance VI orientation in unfamiliar indoor buildings. Three of the instructors are female, and three are male. Instructors' mean age was 50.5 years with 11.9 years standard deviation. Except one, all instructors were experienced instructors who had more than three years O&M teaching experience. The exception was a new instructor who had an experience of less than one year. Interviews were one-to-one, which took between 45-60 minutes for each. There were 22 interview questions, where 12 of them were close-ended and 10 were elaborate open-ended. Instructors were engaged in scenarios about students trying to orient themselves in unfamiliar indoor environments. While instructors may deviate from our scope and express their frustration with other related issues, researchers try to focus on the scope by following the designed questions.

The described O&M best practices were transformed into initial system requirements that match each of the recommendations. For example, a recommendation of the importance of paying attention to the environmental cues such as floor texture can be transformed into a system requirement that says that the system should provide information about floor textures and their types in the open space areas.

VI. PHASE 3: REMOTE ELICITATION WITH PRIMARY STAKEHOLDERS

To elaborate on our gathered initial requirements, we conducted a remote requirement gathering and validation. We wanted to validate our previous initial requirements, and elaborate on them. We designed an online survey, which was intended to validate and extend our gathered requirements. We hosted the survey on an online survey engine that belongs to our university. The online survey was accessible through screen readers to allow visually impaired users to read and interact with them. Users were invited to fill out the survey through social news site ([reddit.com/r/blind](https://www.reddit.com/r/blind)) and online mailing lists. As mentioned before, we invited VI users who are cane users, able to walk unassisted, and not hard of hearing or deaf to participate in the survey.

The survey consists of 27 questions. Questions have different variations such as: demographics, multiple choice, discussion and attitudinal questions. Survey questions discuss many issues such as orientation enhancement in unfamiliar indoor environments, disorientation factors, and input/output user preferences.

In this survey, we received 65 responses. Our participants' mean age was 53.26 with 11.29 years standard deviation. The male-female ratio was about half where 32 were female, 31 male, and two preferred not to answer. In terms of the geographical locations, we had participants from 27 different geographical locations inside and outside the United States (US). Table 2 illustrates the participants' geographical locations. In terms of the visual impairment types, 41 (63.1%) of our sample reported that they are totally blind while 24 (36.9%) of them reported being legally blind.

TABLE I. A SAMPLE OF THE INITIAL REQUIREMENTS BASED ON O&M INSTRUCTORS POINT OF VIEW

| No. | Requirement |
|------|---|
| R1 | The system should provide information to the user about floor texture types in the open spaces |
| R2 | The system should provide the user with information about the surrounding landmarks |
| R2.1 | When a user arrives at an open space area, the system should explain and describe to the user what and where the nearby landmarks are |
| R2.2 | When a user is in motion, the system should provide information about the availability of a landmark |
| R3 | The system should provide information about the open space (how big and how far) |
| R4 | The system should not block user's ears, as a user needs her hearing ability at all times |

TABLE II. NUMBER OF PARTICIPANTS GEOGRAPHICALLY

| ID | Country/State | Count |
|---------------------------------|----------------|-------------------|
| 1 | United States | 56 |
| 2 | Canada | 4 |
| 3 | England | 2 |
| 4 | Australia | 1 |
| 5 | Did not answer | 2 |
| In the United States | | 56 (86.2%) |
| Out of the United States | | 7 (10.8%) |
| Did not answer | | 2 (3.0%) |
| Total | | 65 |

TABLE III. SMARTPHONE OWNERSHIP AND BRAND

| Smartphone | Count (N=65) |
|-----------------------|--------------|
| iOS device (iPhone) | 42 (65.6%) |
| Android phone | 4 (6.2%) |
| Nokia Symbian | 1 (1.5%) |
| Jitterbug | 1 (1.5%) |
| Use smartphone | 45 (69.2%) |
| Do not use smartphone | 16 (24.6%) |
| Did not answer | 4 (6.2%) |

TABLE IV. EXAMPLES OF USER'S REPORTED INFORMATION NEEDS ABOUT LANDMARKS IN THE UNFAMILIAR OPEN INDOOR SPACES

| Information type | Number and percentage (N=65) |
|--|------------------------------|
| Availability and location of reception desk | 60 (92.3%) |
| Bathrooms locations | 59 (90.8%) |
| Elevators and staircases locations | 56 (86.2%) |
| Number of floors | 48 (73.85%) |
| Location of office space | 47 (72.3%) |
| Availability and location of ADA-compliant bathrooms | 11 (16.9%) |
| Did not answer | 4 (6.2%) |

To help us to decide a mobile-based platform for our proposed orientation assistive technology, we asked the users to answer questions regarding their use of smartphones. The majority of our participants use iOS devices (iPhone). Two of our participants reported using more than one smartphone regularly. Table 3 shows participants' ownership of smartphones. Reporting smartphone's ownership and brands can help other software engineers when designing smartphone-based applications for VI users.

To realize user's needs in terms of the needed information, we asked users two different types of questions, those that relate to disorientation factors, and those related to the information needs. Disorientation factors were asked because some users realize their challenges, but may not know what required information should be presented. For example, one of the reported challenges was the difficulties users face to understand what and where are the landmarks around them. This, however, can be transformed into a very key insight to the system requirements that "*a system needs to present information about the surrounding landmarks upon user request.*" Table 4 shows examples of the top information needs for orientation assistive technologies in unfamiliar indoor open spaces to be taken into account.

While a system needs to present information about landmarks in unfamiliar buildings, referencing a landmark can be done using various systems such as cardinal, clock-based, degree, and relative systems. We measure user acceptance to the previous systems in different questions. Figure 3 shows user acceptance to different landmark location reference forms where two of our participants did not answer this question. Learning referencing systems is part of the O&M training; however, we added a description for each system in the survey. The clock-based system was the most preferred way by our

participants to explain any landmark position in relation to users' current position. This however, can give insight when designing a system that involves referencing for this class of users. Figure 4 illustrates the clock-positioning system. In that system, the front of the user's position is 12 O'clock.

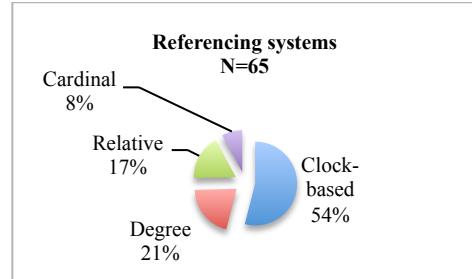


Figure 3. User preferences towards landmark's referencing system

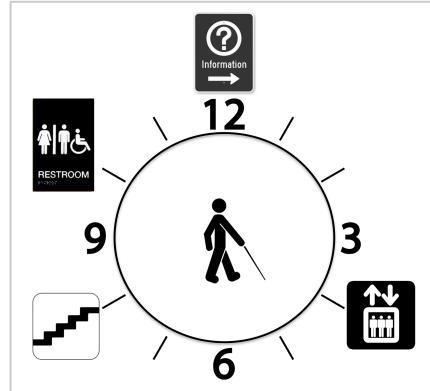


Figure 4. Clock-positioning system explains landmarks around VI user in a clock shape

A. Technology and Interaction Requirements

Through this part of the study, we investigated and elicited user preferences towards the preferred form of technology and user interface, depending on the environment. Users discussed their preference towards voice-based interaction and an easy-to-use interface. Users can query the system using their own voice, and hear the system feedback in a speech form. While voice is the preferred form of interaction, users expressed their need to keep their ears open to pick up environmental sounds while listening to the system feedback. Thus, the use of a special headphone such as bone-conduction technology was recommended. Table 5 shows examples of our top user's initial requirements for the technology and user interface for our proposed orientation assistive technology.

VII. GEOGRAPHICAL REQUIREMENTS DIVERSITY

In our requirement elicitation process, we interviewed and surveyed VI users from different countries around the world. Through the course of the elicitation, we recorded diversity in terms of needs between users from different countries. For example, VI users from Europe have less need for assistance when entering unfamiliar buildings due to the small size of atrium areas, or the structure of buildings. On the other hand, users from the United States (US) reported an urgent need for

assistance when entering unfamiliar buildings. Another example was the need for braille office signs in Australia and New Zealand in contrast to users from US, Canada, and Europe who did not discuss that as a need.

TABLE V. TECHNOLOGY AND USER INTERFACE REQUIREMENTS

| No. | Requirement |
|-----|---|
| R1 | The system should provide voice feedback to the user |
| R2 | When in motion, user can query the system using voice |
| R3 | When not in motion, users can interact with the system using tabs and gestures |
| R4 | To alert user, the system should provide alert information in haptic form (vibration) |
| R5 | The system should provide voice feedback delivered through headphones |
| R6 | User headphones should not block user's ears (use special headphones) |
| R7 | User should be able to adjust system's volume at any time |
| R8 | The system should provide vibration feedback through a vibrating wristband |
| R9 | The system should run on a user's smartphone |
| R10 | The system should work as a hands-free device |

VIII. USER PROFILE

To design our system, we need to understand our targeted user's profile, as the system should accommodate their needs. In this section, we will overview the most important user characteristics that profile our targeted system's users.

A. Demographics

We expect our system to be used by adult VI users, as our domain study included only adults. Another user demographic is the O&M training. We expect that our users have received an O&M training and know how to travel using their cane. Our users are expected to use their cane in addition to our system when entering unfamiliar indoor environments.

B. Visual Impairment

VI users are classified under two visual impairment types: legal and total blindness [3]. As mentioned before, legal blind users have some visual perception of no more than 20/200 in the better eye. This perception can allow them to identify some visual changes in the environment; however, it's not enough for them to travel only using their sight.

C. Ability to Hear

Due to our intended user interface design, we decided to consider VI users who are not hard of hearing or deaf. The reason behind this decision is that deaf-blind users communicate with systems using only a figure-braille method [15]. Additionally, deaf-blind users were not included in our domain understanding.

D. Cane Users

VI persons can vary from people who can travel unaided or with aid. Users who travel with aid use two aids: canes and guide-dogs. In our domain understanding elicitation study, we discovered less need towards an additional assistive tool to the VI users who use guide-dog in contrast to cane users. This led our decision to specify our system to the cane users.

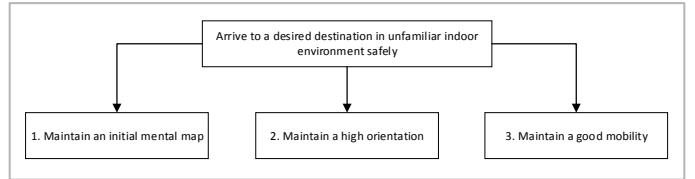


Figure 5. A high-level analysis of user's goals that was conducted using GDTA analysis.

E. Ability to Walk Unassisted

The ability of VI users to walk unassisted when orienting in unfamiliar indoor environments is one of the characteristics that we selected for our system's users. The reason behind selecting it was the desire of conducting controlled experiments to test the system at a later stage. Our potential controlled experiment environments are not suitable to accommodate VI users who are not able to walk. Also, VI users who are assisted by humans have no use for an orientation tools as their assistants can lead them through different environments.

IX. SITUATION AWARENESS REQUIREMENTS

For the purpose of our design, we elicited SA requirements in parallel with system requirements. As mentioned before in the methodology section, SA requirements can be conducted through identifying user goals. We used GDTA as our method to elaborate on each user goal and divided each goal into sub-goals as needed. Also, we've attached to each goal the associated user decision and the required information to make that decision. Information required to maintain each identified goal can be obtained either from the environment or from a system. Our analysis was conducted using our elicited requirements from our interview studies: VI users and O&M instructor interviews.

SA requirements can help in forming design decisions as they show what information can be presented to the user and when. Some systems can present unneeded information in different situations, which might overwhelm the user.

When visiting an unfamiliar building, VI users have a common goal of *"arriving to a desired destination in an unfamiliar indoor environment safely."* Using this goal, we elicited, and elaborated with users to identify what are the sub-goals, and what information is needed to achieve each. Figure 5 shows a high-level elaboration of user goals when entering unfamiliar indoor environments.

A. User's Initial Mental Map

Building user's mental map at the beginning of orientation tasks in any unfamiliar buildings is crucial to user's performance and safety. VI users use their prior knowledge such as a descriptive information about the environment to build an initial mental map about the environment. Prior knowledge, however, is not enough for VI users to have a good mental map and be able to orient themselves in such environments. Through our interview's elaboration, we explored with our users and instructors different scenarios

when a VI user enters an unfamiliar indoor environment. Using our gathered data, we've conducted an in-depth GDTA analysis that presents what information is needed to assist VI user's initial mental map when entering unfamiliar indoor environments. Figure 6 shows a brief view of goal (1) analysis. Note that the provided sub-goals are not ordered. Figure 9 shows the SA requirements blocks that explain each of the information needs.

Determining and understanding the surrounding important landmarks (goal 1.1) are so crucial as they help VI users to visualize the surrounding environment and make their travel decisions based on the availability of landmarks. We described what are the important landmarks earlier in the domain understanding section.

One of the goals that help VI users to maintain a good initial mental map of the environment is the sense of direction (goal 1.2). Users need to know their direction, as they need to maintain their understanding of the changes in their direction when they travel in indoor environments. This is an important step, as they need to reverse the order of the previous directions to know their way back when exiting the building or any part of it.

As VI users need to travel through unfamiliar indoor environments, they need to plan their path (goal 1.3). To plan their path, users need to use their prior knowledge of the environment and decide the easiest path to follow. Planning user's path, however, needs environmental information such as what landmarks are available in the environment and where they are located. Depending on the environmental information available to the user, a user can plan an initial path to follow when traveling.

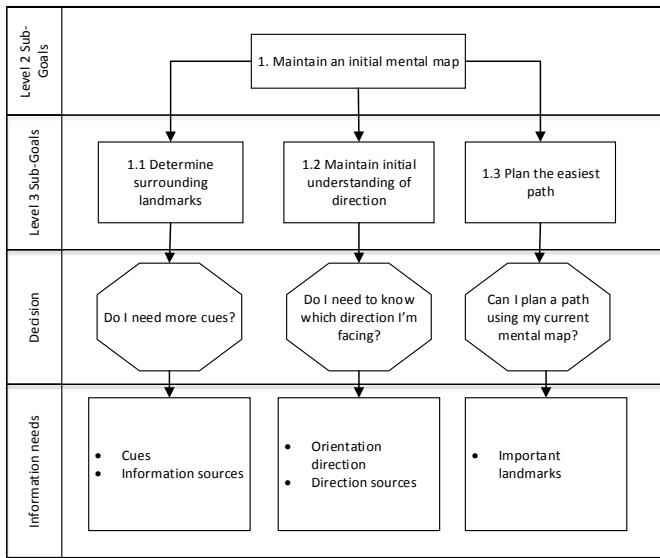


Figure 6. A Goal-Directed Task Analysis for goal number (1)

B. User's Orientation

The second core sub-goal of VI user's goals when entering unfamiliar indoor environments is to maintain a high orientation. This is an important goal, as users need to be able to change their path due to any changes in the environment.

Using our interviews' data, we divided this goal into three sub-goals. Figure 7 illustrates our GDTA analysis for "maintaining a high orientation" goal. Figure 9 shows SA requirements blocks of information needs listed here.

In the previous section, we mentioned the importance of the initial understanding of the direction that a user is facing. This is extended here to understand the direction that a user is facing when traveling indoors (goal 2.1). This goal is important, as a user need to maintain a relational path of the environment in order to be able to travel back and exit the building or any part of it.

While a user is traveling through an unfamiliar indoor environment, the initial user's mental map is not enough. A user needs to maintain and update the initial mental map to be able to travel efficiently. Goal 2.2 illustrates the user desire to understand the environment in a continuous manner. This means that a user needs to acquire information about landmarks' locations, in addition to understanding where the previous landmarks are that a user passed by, and to predict what landmarks will be in the user's way.

Although VI users need to maintain their path planning when traveling in unfamiliar indoor environments, they also need to maintain their safety. Detecting obstacles in the user's way (goal 2.3) is an important goal to maintain a safe orientation. Users need to be able to identify obstacles on their way using their assistive tools.

C. User's Mobility

While orientation is an important factor in the VI travel in unfamiliar indoor environments, maintaining that orientation when traveling is yet another goal that VI users need to maintain. Figure 8 shows our conducted GDTA for goal 3: "maintain a good mobility." Figure 9 shows the SA requirements blocks that can explain each of users' information needs.

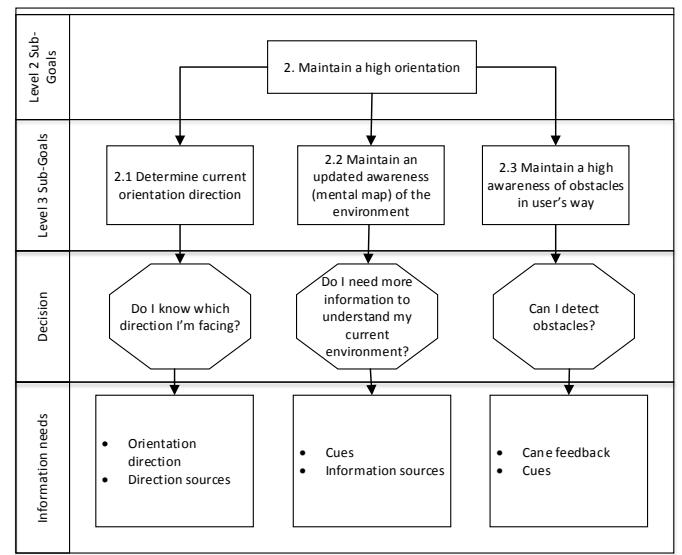


Figure 7. A Goal-Directed Task Analysis for goal number (2)

User's need to avoid anything that may affect their indoor travel such as dead-ends. Goal 3.1 illustrates user's desire to

avoid any dead-end paths. Information about a path, such as environmental cues, obstacles that cannot be traveled around and closed doors, can provide an indicator to the VI user to change the travel path immediately.

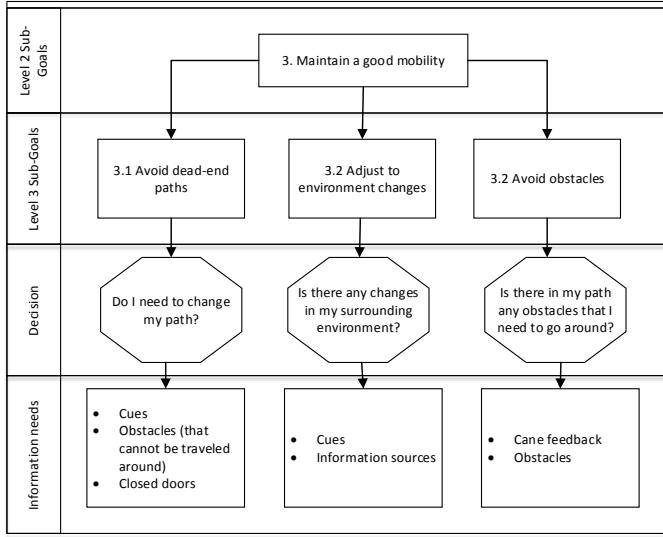


Figure 8. A Goal-Directed Task Analysis for goal number (3)

Adjusting to environmental changes is another crucial user's goal (goal 3.2) that the VI user needs to maintain. Pedestrian traffic, for example, can lead users to decide changing their path when traveling.

As mentioned before, users need to be able to detect obstacles when they are traveling unfamiliar indoor environments. However, this is not enough to maintain user's safety. Avoiding obstacles is another safety goal (goal 3.3) when user is in motion. This goal helps in maintaining user's safety as well as knowing obstacles in the user's way.

X. LESSONS LEARNED

Through our requirement elicitation stage, we learned lessons that could help the future elicitation studies that work in the field. Our lessons learned can be classified into two categories: first, lessons learned from the elicitation process; second, lessons learned from the knowledge elicitation and the combination of different stakeholders' views.

A. Elicitation Process

When working with non-standard user population, software engineers need to take into account that the standard ways of conveying ideas may not always be ideal. In this work, we faced three elicitation challenges that can be categorized as the following:

- **Domain challenges:** since our intended solution is aimed for the domain of visual impairment, we had to learn how VI individuals travel indoors and what their needs are when traveling unfamiliar environments. This knowledge, however, should be combined with domain experts' opinion from the same domain. Domain experts can provide a different

view of the same needs. They also help in elaborating upon the previous elicited requirements.

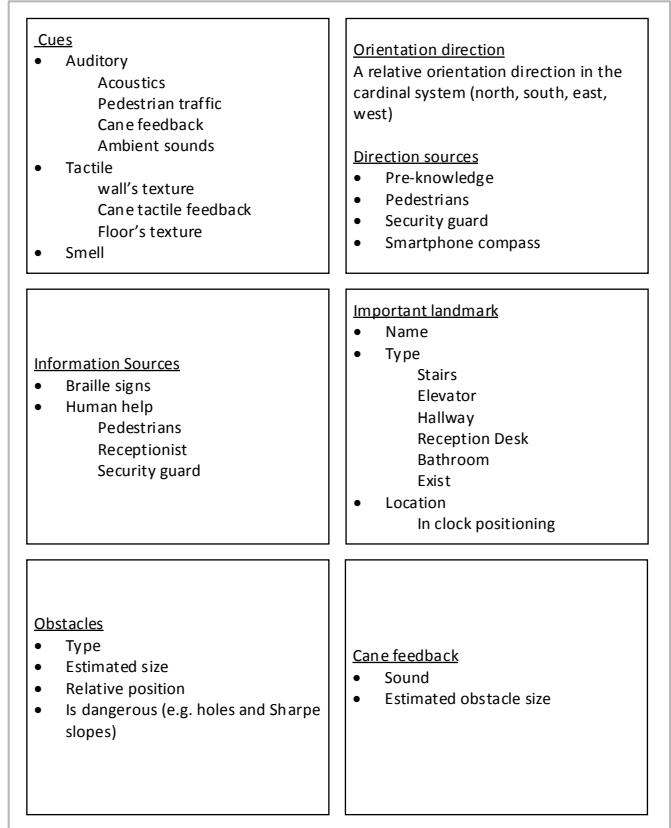


Figure 9. SA requirements blocks

- **Remote elicitation:** remote elicitation can provide great feedback to software engineers who want their design to work in different geographical areas. In our elicitation study, we sought VI participants from different geographical locations worldwide. This helped us to formulate the system needs based on different views. Accommodating the preferred time for interviews in remote locations was a challenge. In some countries far east, it was difficult to find a time that suits participants and researchers. Remote elicitation process could involve working out of the normal working hours.
- **Non-visual elicitation:** eliciting requirements from VI stakeholders indicates that visual aids cannot be used. To work this out, we developed a scenario-based explanation of features and properties that we discuss with users.

B. Combining Different Stakeholders' Views

In some cases, VI stakeholders and domain experts do not share the same opinions when it comes to system features and interactions. In this work, we faced a challenge of eliciting requirements from the primary stakeholders that conflict secondary stakeholders' (domain experts) point-of-view. An example of this is when some users told us that they would like to hold their smartphones and communicate with an

embedded orientation assistive technology that resides in them. On the other hand, O&M instructors stressed that holding the smartphone when traveling may result in unwanted consequences such as injuries. To resolve these conflicts in the initial requirements, we further asked the users in the online survey about the conflicting requirements while stressing the issues that were raised by the secondary stakeholders. This provided a good context for us to elicit user's response to such conflicting requirements.

XI. CONCLUSION AND FUTURE WORK

VI users strive to understand unfamiliar indoor environments when they travel through them. Maintaining a high SA level of VI users in such environments is crucial as it helps to enhance their orientation abilities and safety. In this paper, we discussed our elicitation process and results to gather user and knowledge requirements for an SA orientation assistive technology dedicated to the VI users. We discussed our process in eliciting requirements from remote stakeholders distributed among different geographical locations. We used interviews and surveys to elicit our initial requirements. Also, we showed the vital environmental information in the unfamiliar indoor environments such as landmark locations that need to be taken into account when designing an assistive orientation technology for the VI. Our work provides a foundation for SA applications that are dedicated to support VI orientation indoors.

Our next step is to validate and finalize our users' requirements using a participatory design (PD) approach [6]. Using this method, we will involve four VI users in the requirement and design processes. In PD, users along with software engineers can perform many software engineering activities, such as requirements analysis, design, and prototyping. We will perform the PD with our users in an iterative manner to enhance our final system design.

ACKNOWLEDGMENT

The authors would like to express their gratitude to all of the VI individuals and O&M instructors who dedicated their time to participate in this research. Also, the first author would like to thank King Saud University in Riyadh, Saudi Arabia for their provided scholarship.

REFERENCES

- [1] A. Strauss and J. Corbin, *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory*, vol. 3. 2455 Teller Road, Thousand Oaks California 91320 United States: SAGE Publications, Inc., 2008, p. 379.
- [2] W. H. Jacobson, *The Art and Science of Teaching Orientation and Mobility to Persons with Visual Impairments*. New York: AFB Press, 1993.
- [3] "Disability Evaluation Under Social Security: special senses and speech - adult." [Online]. Available: <http://www.ssa.gov/disability/professionals/bluebook/2.00-SpecialSensesandSpeech-Adult.htm>. [Accessed: 19-June-2014].
- [4] "Visual impairment and blindness: key facts." [Online]. Available: <http://www.who.int/mediacentre/factsheets/fs282/en/>. [Accessed: 13-Feb-2014].
- [5] M. R. Endsley, *Designing for Situation Awareness: An Approach to User-Centered Design*, Second Edi. Boca Raton, FL, USA: CRC Press, Inc., 2011.
- [6] M. J. Muller, "Participatory design: The third space in HCI," in *Human-Computer Interaction Handbook*, J. Jacko and A. Sears, Eds. L. Erlbaum Associates Inc., 2003, pp. 1051–1068.
- [7] B. Crandall, G. Klein, and R. Hoffman, *Working minds: A practitioner's guide to cognitive task analysis*. Cambridge, MA: MIT Press, 2006.
- [8] T. Strohotte, H. Petrie, V. Johnson, and L. Reichert, "MoBIC: user needs and preliminary design for a mobility aid for blind and elderly travellers," *Computer (Long. Beach. Calif.)*, vol. 1, pp. 348–351, 1995.
- [9] V. Johnson and H. Petrie, "Travelling safely: the problems and concerns of blind pedestrians," *Br. J. Vis. Impair.*, vol. 16, no. 1, pp. 27–31, Jan. 1998.
- [10] A. M. O'Neill, H. Petrie, G. Lacey, N. Katevas, M. A. Karlson, P. Engelbrektsson, B. Gallagher, H. Hunter, and D. Zoldan, "Establishing initial user requirements for PAM-AID: a mobility and support device to assist frail and elderly visually impaired persons," in *Improving the Quality of Life for the European Citizen*, I. P. Porrero and E. Ballabio, Eds. IOS PRESS, 1998, pp. 292–295.
- [11] I. Rafael, L. Duarte, L. Carriço, and T. J. V. Guerreiro, "Towards ubiquitous awareness tools for blind people," in *BCS HCI*, 2013, p. 38.
- [12] M. Miao, M. Spindler, and G. Weber, "Requirements of Indoor Navigation System from Blind Users," in *Information Quality in e-Health SE - 48*, vol. 7058, A. Holzinger and K.-M. Simonic, Eds. Springer Berlin Heidelberg, 2011, pp. 673–679.
- [13] J. Lazar, J. H. Feng, and H. Hochheiser, *Research Methods in Human-Computer Interaction*. Wiley Publishing, 2010.
- [14] P. Engelbrektsson, I. C. M. Karlsson, B. Gallagher, H. Hunter, H. Petrie, and A.-M. O'Neill, "Developing a navigation aid for the frail and visually impaired," *Univers. Access Inf. Soc.*, vol. 3, no. 3–4, pp. 194–201, 2004.
- [15] M. Hirose and T. Amemiya, "Wearable Finger-Braille Interface for Navigation of Deaf-Blind in Ubiquitous Barrier-Free Space," in *Proceedings of the Tenth International Conference on Human-Computer Interaction*, 2003, pp. 1417–1421.
- [16] M. R. Endsley, "Design And Evaluation For Situation Awareness Enhancement," *Hum. Factors Ergon. Soc. Annu. Meet. Proc.*, pp. 97–101, 1988.
- [17] A. Sutcliffe, S. Fickas, and M. M. Sohlberg, "Personal and contextual requirements engineering," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005, pp. 19–28.
- [18] J. T. Hackos and J. C. Redish, *User and Task Analysis for Interface Design*. New York, NY, USA: John Wiley & Sons, Inc., 1998.

Supporting Early Decision-Making in the Presence of Uncertainty

Jennifer Horkoff*, Rick Salay†, Marsha Chechik†, and Alessio Di Sandro†

*Department of Information Engineering and Computer Science, University of Trento, Italy

horkoff@disi.unitn.it

†Department of Computer Science, University of Toronto, Toronto, Canada

{rsalay,chechik,adisandro}@cs.toronto.edu

Abstract—Requirements Engineering (RE) involves eliciting, understanding, and capturing system requirements, which naturally involves much uncertainty. During RE, analysts choose among alternative requirements, gradually narrowing down the system scope, and it is unlikely that all requirements uncertainties can be resolved before such decisions are made. There is a need for methods to support early requirements decision-making in the presence of uncertainty. We address this need by describing a novel technique for early decision-making and tradeoff analysis using goal models with uncertainty. The technique analyzes goal satisfaction over sets of models that can result from resolving uncertainty. Users make choices over possible analysis results, allowing our tool to find critical uncertainty reductions which must be resolved. An iterative methodology guides the resolution of uncertainties necessary to achieve desired levels of goal satisfaction, supporting trade-off analysis in the presence of uncertainty.

I. INTRODUCTION

During the process of eliciting requirements, it is common to encounter uncertainties, including gaps in domain knowledge, disagreements between stakeholders, or uncertainty over requirement details. Early requirements elicitation typically uncovers a large space of alternative requirements and conflicting needs, making it necessary to support decision-making over alternatives in order to find acceptable trade-offs between conflicting requirements. Given the wide scope of early system analysis, it may not be feasible to resolve all uncertainty before decision-making occurs. Ignoring uncertainty forces implicit and premature decisions over the space of early requirements. Thus, methods and tools to (a) support early decision-making and trade-off analysis and (b) do so in the presence of uncertainty are needed. To support these needs, we make use of existing, established Requirements Engineering (RE) techniques: goal modeling and associated analysis [11], and the *MAVO* framework for formally capturing and reasoning over model uncertainty [20].

Goal models (e.g., NFR [3], KAOS [4], i* [24]) have been widely studied in RE as a means to explicate system goals and high-level requirements. The ability to facilitate the analysis of alternative requirements makes goal models especially powerful for RE. For example, “what if?” analysis [3], [7], [11] uses the effects on and relationships between goals to determine the (degrees of) goal satisfaction resulting from a selection of alternative requirements. Such analysis helps modelers select

a viable set of requirements, making acceptable trade-offs between user needs.

Current goal model analysis procedures evaluate the effects of alternative requirements on user goals but do not account for model uncertainties discovered at design time. Much focus has been placed on capturing and reasoning over vague goals, using the softgoal concept to capture goals without clear-cut criteria for success. However, in addition to being vague, goals (and associated relationships) can be uncertain, for example, we might be uncertain about the presence or absence of an element. Ignoring uncertainty, i.e., treating the element as present, may lead us to eliminate viable alternatives or accept alternatives which do not sufficiently satisfy key goals.

The *MAVO* framework [20] has been introduced in order to explicitly and formally capture uncertainty in the contents of models. *MAVO* is language-independent, and our previous work, [18], [19], allowed us to record model changes, including change rationale, checking that these changes reduce uncertainty, and to propagate uncertainty reductions across traceability links between different models. In this work, we take the semantics of the target language (i*) into account, allowing us to determine the satisfaction of goals, evaluate alternative solutions, make tradeoffs and find satisfactory solutions – even in the presence of model uncertainty.

Specifically, we facilitate analysis and decision-making in the presence of uncertainty by integrating goal model analysis with uncertainty expressed using *MAVO*. We describe a semi-automated method which finds sets of possible goal satisfaction analysis results, accounting for all of the ways in which model uncertainty may be resolved. Our approach determines which uncertainties must be resolved to achieve desired levels of goal satisfaction, allowing for targeted domain elicitation. We provide a methodology which supports an iterative reduction of uncertainty, moving towards “concrete” models in which uncertainties are resolved and desired levels of goal satisfaction are achieved. Thus, we allow users to make decisions over the space of early requirements even in the presence of uncertainty.

A variety of goal modeling languages and analysis procedures have been introduced (e.g., [3], [24], [4], [7]). In order to make our contribution concrete, we selected a particular language, i* [24], and a particular method for qualitative analysis [11]. However, our approach can be generalized to

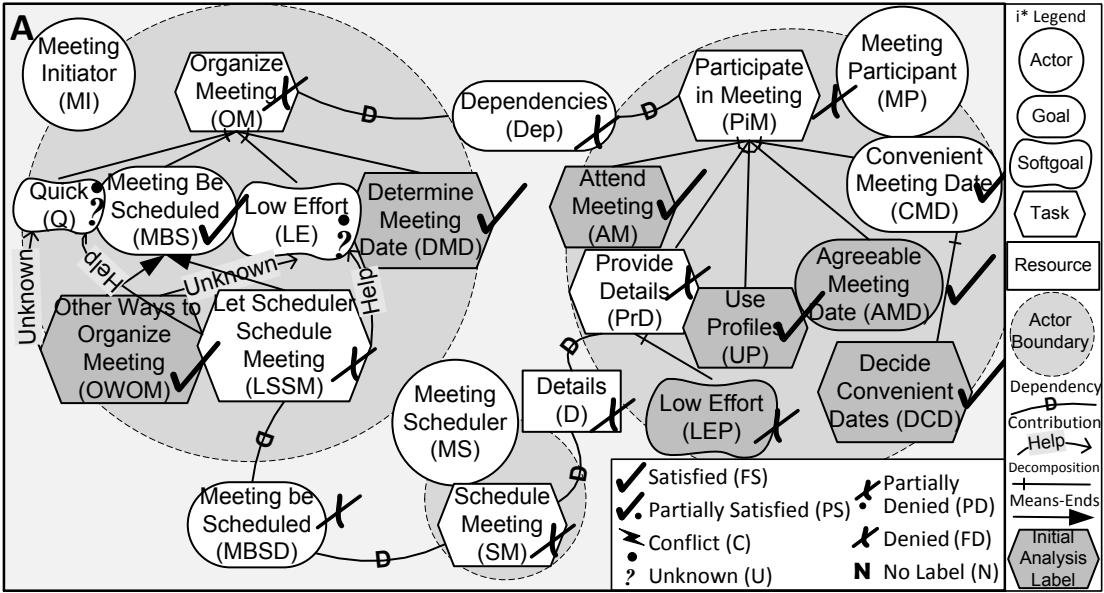


Fig. 1. A goal model for the Meeting Scheduler Example with analysis of the OWOM alternative.

any goal-oriented language with formal semantics. We do not model the intrinsic, run-time uncertainty of the environment, but the design-time uncertainty of the modeler, which can be eventually resolved. This type of uncertainty is best captured *possibilistically* rather than *probabilistically* as the statistics needed for the latter is typically not available at design time.

The rest of this paper is organized as follows: Sec. II describes a motivating scenario and notation background using the Meeting Scheduler example. We provide the necessary formal background and notation in Sec. III. Sec. IV describes how to compute and interpret analysis labels over uncertain models. Sec. V describes a methodology for analysis of uncertain models, including finding suggested uncertainty refinements given analysis choices. Sec. VI describes our tool support and experiences. We compare our approach with related work in Sec. VII. Sec. VIII summarizes the paper and discusses future work.

II. BACKGROUND AND MOTIVATING SCENARIO

We motivate our approach using a well-known meeting scheduler example¹. In this scenario, a Meeting Initiator has a variety of ways to schedule meetings, including using a (semi)automated Meeting Scheduler, and must chose between scheduling alternatives. However, in the early analysis of this scenario, we are uncertain about several aspects of the domain. What are some other ways for the Meeting Initiator to schedule a meeting? What information is needed from the Meeting Participant? Who determines possible meeting dates? Despite these uncertainties, we need to make decisions over the space of scheduling alternatives. We ask: Can we evaluate whether or not an alternative is viable even in the presence of uncertainty? and What uncertainties must be resolved in order to facilitate our decision-making process?

¹Parts of this scenario have appeared in previous work, e.g., [24], [18].

i* Language. Fig. 1 depicts an i* model, model A for the meeting scheduler capturing a simplistic view of the goals, actors and dependencies for this system. It does not yet explicitly capture our uncertainties concerning the scenario. The social aspect of i* is represented by *actors*, e.g., Meeting Initiator (MI for short, see Fig. 1 for shorthands), and Meeting Participant (MP). Actors depend upon each other for the accomplishment of *tasks*, e.g., Provide Details (PrD), and Determine Meeting Date (DMD), the provision of *resources*, e.g., Details (D), the satisfaction of *goals*, e.g., Meeting be Scheduled (MBS) and *softgoals*, goals without clear-cut satisfaction criteria, e.g., Low Effort (LE). Together, these are called *intentions*. Actor “boundaries” contain the intentions explicitly desired by these actors, e.g., MP wants to achieve Participate in Meeting (PiM).

The interrelationships between the intentions are depicted via three types of links: *decomposition*, representing intentions necessary to accomplish a task, e.g., to achieve PM the MP actor must satisfy AM, PrD, and satisfy the other three decomposition intentions; *means-end*, representing alternative tasks which can accomplish a goal, e.g., Other Ways to Organize Meeting (OWOM) or Let Scheduler Schedule Meeting (LSSM) will satisfy MBS; and *contribution*, showing the effects of softgoals, goals, and tasks on softgoals, e.g., LSSM helps Quick (Q). Positive/negative contributions representing evidence sufficient to satisfy/deny a softgoal are represented by *Make/Break* links, respectively. Weaker contributions are represented by *Help/Hurt* links. *Unknown* contribution links represent the presence of evidence with yet unknown polarity.

Goal Model Analysis. Model A allows the analyst to capture requirement options, and existing analysis procedures, e.g., [11], allow her to evaluate such alternatives. The analysis procedure starts with a question of the form “How effective is an alternative w.r.t. the goals in the model?”. In our example,

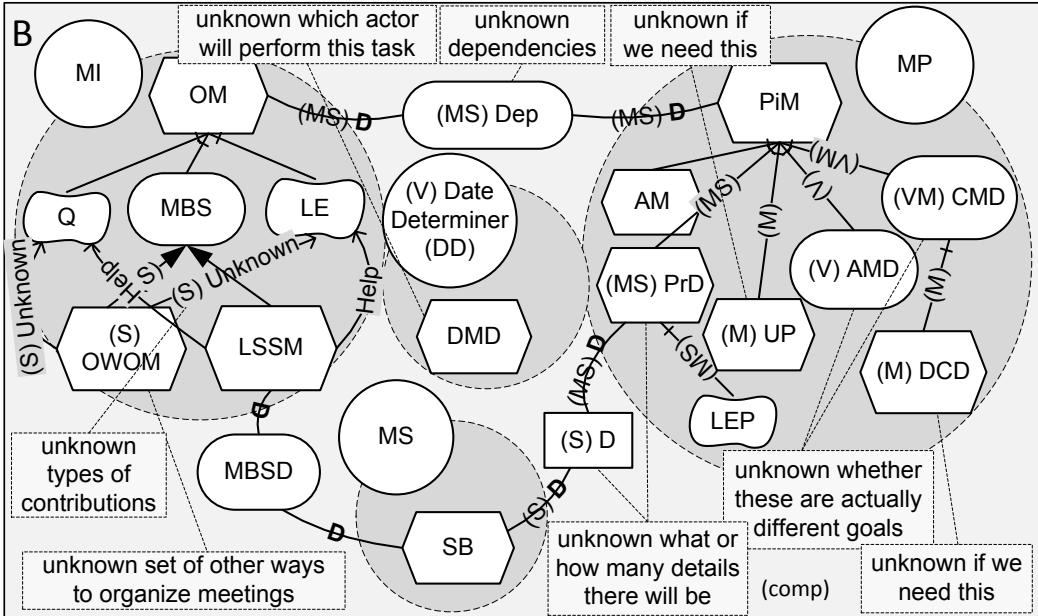


Fig. 2. A goal model for the Meeting Scheduler Example annotated with uncertainties informally (dashed notes) and formally (annotations).

there are two alternative ways to satisfy **MBS**: **LSSM** and **OWOM**. If we want to find the effects of selecting **OWOM**, we place *initial labels*, satisfying the task **OWOM** and denying **LSSM** in model **A**. For completeness, other initial labels, reflecting a more detailed analysis question, are chosen (shaded intentions in model **A**).

Initial labels are iteratively propagated through links using propagation rules for decomposition, means-ends, dependency, and contribution links, resulting in labels for other model elements, representing the cumulative level of positive or negative evidence. The propagation continues until no new labels can be produced. In our example, initial labels for **OWOM** and **LSSM** are propagated through the means-end (OR) link. By this rule, **MBS** is (fully) satisfied (*FS* or \checkmark , see legend in Fig. 1). Similarly, incoming labels for **PiM** in **MP** are propagated through the (AND) decomposition, this time selecting the minimum label, fully denied (*FD*).

Since there is a dependency relationship between **D** and **PrD**, a denied value for the latter implies a denied value for the former. Propagation through contribution links uses rules adopted from [3]. In order to capture partial evidence propagated through contribution links, goal model analysis captures *partial positive* or *negative* evidence towards goal satisfaction, via *partially satisfied* (*PS*) and *partially denied* (*PD*) labels, respectively. For example, the *FD* label for **LSSM** propagates a *PD* label through the *Help* link to **Q** and **LE**. Similarly, the *Unknown* label (*U*) represents the presence of evidence which could be positive or negative. For example, the *FS* value for **OWOM** propagates *U* through the *Unknown* links. In this case, *U* represents a missing piece of evidence that should be addressed by users.

As softgoals often have multiple incoming contribution links, multiple labels might apply at once, e.g., **Q** is both *U*

and *PD*. Such cases can be combined into a single label for subsequent propagation; possibly making use of the *Conflict* label, (*C*) indicating the presence of positive and negative evidence of roughly the same strength. In this paper, we combine evidence automatically: we select *U* when present, or *C* when evidence is both positive and negative, and otherwise select the minimum (most pessimistic) label. When an intention is the target of both a dependency link and another type of link (decomposition, means-ends, contribution), the propagation results from both types of links are combined via AND (taking the minimum label). The *N* (no label) symbol is used to explicitly indicate the absence of other labels.

Domain experts and analysts can determine whether the analysis results are sufficient or *viable*. In our example, the analysis indicates that the **MI** actor is unable to satisfy **OM** (*FD*), so **OWOM** is a non-viable alternative. In this case, the modeler would analyze other model alternatives (**LSSM**) or perform additional elicitation to find further alternatives.

Model Uncertainty. As mentioned, the scenario involves several points of uncertainty. We summarize this uncertainty using free-form notes in model **B** in Fig. 2. We can use the *MAVO* method described in [18] to formally express the uncertainties in our model, shown in the same figure.

MAVO uses four types of annotations, each adding support for a different type of uncertainty in a model. The *May* annotation allows us to express uncertainty about the presence of an element in a model. Annotating it with *M* indicates that it “may exist”; otherwise, it “must exist”. E.g., in model **B**, we are unsure if the **MP**’s task **DCD** is present in the model. Uncertainty is reduced by removing the *May* annotation or eliminating the element altogether.

The *Abs* annotation allows a modeler to express uncertainty about the number of elements in the model. An *S* represents

TABLE I
ANALYSIS QUESTIONS OVER UNCERTAIN GOAL MODELS.

| | |
|-----------|---|
| Q1 | What are the analysis results given a particular analysis alternative in the goal model, considering model uncertainties? |
| Q2 | Can viable choices be made over the set of results from Q1 ? |
| Q3 | Can viable choices be achieved simultaneously? If so, find example uncertainty resolutions that achieve choices. |
| Q4 | Given choices, what uncertainty reductions are forced? How can we target elicitation? |
| Q5 | Are uncertainty reductions suggested by Q3 reasonable given the domain? If not, find further uncertainty reductions. |

TABLE II
SELECTED PROPAGATION RULES.

| Link Type | | Original Rule | Example <i>MAVO</i> Rule |
|---------------|--|---|---|
| Dependency | | $(v \in V) v(i_s) \Rightarrow v(i_d)$ | $\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Dep}(t, g) \wedge \text{FS}(g)) \Rightarrow \text{FS}(t)$ |
| Decomposition | | $(\bigwedge_{j=1}^n \text{FS}(i_j)) \Rightarrow \text{FS}(i_d)$ \dots $(\bigvee_{j=1}^n \text{FD}(i_j)) \Rightarrow \text{FD}(i_d)$ | $\forall t_1 \dots t_n : \text{Task}, g : \text{Goal} \cdot (\text{Decomp}(t_1 \dots t_n, g) \wedge \bigwedge_{j=1}^n \text{FS}(t_j)) \Rightarrow \text{FS}(g)$ \dots $\forall t_1 \dots t_n : \text{Task}, g : \text{Goal} \cdot (\text{Decomp}(t_1 \dots t_n, g) \wedge \bigvee_{j=1}^n \text{FD}(t_j)) \Rightarrow \text{FD}(g)$ |
| Contribution | | $(c = \text{Make}) \text{FS}(i_s) \Rightarrow \text{FS}(i_d)$ $(c = \text{Help}) \text{FS}(i_s) \Rightarrow \text{PS}(i_d)$ \dots $(c = \text{Unk}, v \in V) v(i_s) \Rightarrow \text{U}(i_d)$ $(c \in \{\text{Make}, \text{Help} \dots\}) \text{U}(i_s) \Rightarrow \text{U}(i_d)$ | $\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Make}(t, g) \wedge \text{FS}(t)) \Rightarrow \text{FS}(g)$ $\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Help}(t, g) \wedge \text{PS}(t)) \Rightarrow \text{PS}(g)$ \dots $\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Unknown}(t, g) \wedge v(t)) \Rightarrow \text{U}(g)$ $\forall t : \text{Task}, g : \text{Goal} \cdot (c(t, g) \wedge \text{U}(t)) \Rightarrow \text{U}(g)$ |

a “set”; omitting the annotation means that the element is “particular”. E.g., the *s*-annotated operation **OWOM** in actor **MI** in model **B** indicates that there are some such operations but they are as yet unknown. Uncertainty is reduced by elaborating the content of *s* elements into a set of (possibly annotated) elements.

The *Var* annotation allows a modeler to express uncertainty about the distinctness of individual elements in the model by annotating an element as a “variable” (*V*); omitting the annotation means that the element is a “constant”. Uncertainty is reduced by merging variable elements with constants or other variables. E.g., while it is known that task **DMD** should be in model **B**, it is not yet clear which actor should perform it. Yet it must be assigned to *some* actor for *i** well-formedness. We thus put the task in a *V*-annotated actor, allowing it to merge with other actors and eventually be assigned to a “real” (constant) actor.

The *OW* annotation allows a modeler to explicitly state whether her model is incomplete (i.e., can be extended) (*INC*) or complete (the default). Here the annotation is at the level of the entire model rather than individual elements. Our model **B** is complete.

Analysis with Uncertainty. We want to be able to make early decisions over alternative requirements, even in the presence of uncertainty. Given our explicit recording of uncertainties in the modeling process, we want to know how their presence affects analysis results (Fig. 1), and how such uncertainties can be used to guide analysis and elicitation. Specifically, (**Q1**) how do the uncertainties in model **B** affect analysis results in model **A**? For example, if we are not sure about the presence of **PrD** (labelled with *FD*), perhaps the **MP** actor may be able to satisfy **PiM** (i.e., this task could be *FS*). If the **MP** actor can satisfy **PiM**, then the satisfaction of **OM** in the **MI** actor may actually be unknown (*U*). We wish to see all of the possible labels given possible uncertainty reductions in the model.

Given the answer to **Q1**, we wish to answer additional questions as listed in Table I. The rest of this paper describes a

method which takes uncertainty into account when analyzing goal models, allowing us to answer these questions.

III. PRELIMINARIES

In this section, we give the necessary formal background on goal model analysis and uncertainty via the *MAVO* framework.

A. Goal Model Analysis

The *forward propagation algorithm* [11] begins by applying initial labels corresponding to the analysis question to the model, queuing the labels. All labels in the queue are propagated through outgoing links, with multiple incoming labels resolved using human judgment as in [11] or, in our case, rules as described in Sec. II. Propagation ends when the label queue is empty (see [11] for a discussion of termination).

The above procedure uses six predicates over model elements corresponding to the six analysis labels ($\vee = \{\text{FS}(), \text{PS}(), \text{C}(), \text{U}(), \text{PD}(), \text{FD}()\}$), where the predicate holds if the label applies.

Initial Analysis Labels. A selected subset of intentions within a goal model are assigned initial analysis labels, e.g., $\text{FS}(\text{OWOM})$ and $\text{FD}(\text{LSSM})$ in Fig. 1.

Propagation Constraints. The procedure provides rules in order to facilitate a standard propagation of labels through goal model relationships (links). The Original Rule column of Table II presents selected propagation rules for the dependency, decomposition, and contribution relationships depicted in the Link Type columns. Dependency links propagate evidence unchanged, decomposition combines evidence using “AND” (minimum), while means-ends combines evidence using “OR” (maximum). The following order between labels is used: $(\text{FS} \geq \text{PS} \geq \text{U} \geq \text{C} \geq \text{PD} \geq \text{FD})$. Furthermore, $\text{FS}(i) \Rightarrow \text{PS}(i)$ and $\text{FD}(i) \Rightarrow \text{PD}(i)$, where *i* is an element in the model.

Means-end rules (omitted from Table II) are derived from decomposition rules by replacing \wedge with \vee and vice versa. Contribution links can weaken or negate evidence, or make evidence unknown, depending on the incoming label, $(v(i_s))$,

and the type of contribution link. For example, the second to last row of the table says that for an *Unknown* (Unk) contribution link, regardless of the incoming analysis label ($v(i_s)$), the *U* label is propagated ($U(i_d)$ holds).

B. Capturing Uncertainty with MAVO

MAVO [20] is an approach for adding uncertainty information as annotations in models. Although *MAVO* can be applied to any type of goal model, this exposition uses i^* models. A *MAVO* i^* model contains annotations on the model elements that together represent a set of different possible *concrete* (i.e., ordinary) i^* models that would resolve the uncertainty:

Definition 1 (MAVO i^* model): A *MAVO* i^* model M consists of an i^* base model, denoted $bs(M)$, and a set of annotations. $[M]$ denotes the set of i^* models called the *concretizations* of M . M is called *consistent* iff it allows at least one concretization, i.e., $[M] \neq \emptyset$.

Formalizing MAVO. We now describe how *MAVO* annotations formally characterize a set of i^* models – it is the foundation of the method for encoding and automatically reasoning with *MAVO* models. The i^* metamodel represents the set of well-formed i^* models and can be expressed as a First Order Logic (FOL) theory.

Definition 2 (i^* metamodel): The i^* metamodel is an FOL theory $T = \langle \Sigma, \Phi \rangle$, where Σ is the *signature* with sorts representing the entity types (e.g., Actor, Intention, Task) and predicates representing the relations (e.g., Task decomposes Goal); Φ is a set of sentences representing the i^* well-formedness constraints (e.g., multiplicities, dependencies between actors). The models that conform to T are the finite FO Σ -structures that satisfy Φ according to the usual FO satisfaction relation. We denote this set of models by $Mod(T)$.

Like the i^* metamodel, a *MAVO* i^* model also represents a set of models and thus can also be expressed as an FOL theory. Specifically, for a *MAVO* i^* model M , we construct a theory $FO(M)$ s.t. $Mod(FO(M)) = [M]$. We illustrate the construction of $FO(\mathbf{B})$ for *MAVO* i^* model \mathbf{B} in Fig. 2.

(1) Let $\mathbf{G} = bs(\mathbf{B})$ be the base model of model \mathbf{B} . We define a new *MAVO* i^* model \mathbf{B}_G with \mathbf{G} as its base model and its sole concretization, i.e., $bs(\mathbf{B}_G) = \mathbf{G}$ and $[\mathbf{B}_G] = \{\mathbf{G}\}$. We call \mathbf{B}_G the *ground* model of \mathbf{B} .

(2) To construct the FOL encoding of \mathbf{B}_G , $FO(\mathbf{B}_G)$, we extend the i^* metamodel to include a unary predicate for each

Σ_G has unary predicates $MP(Actor)$, $AM(Task)$, ..., and binary predicates $AMinMP(Task, Actor)$, ...
 Φ_G contains the following sentences:
 $(Complete) (\forall x : Actor \cdot MP(x) \vee MS(x) \vee DD(x) \vee \dots) \wedge$
 $(\forall y : Task, x : Actor \cdot in(y, x) \Rightarrow (AMinMP(y, x) \vee \dots)) \wedge \dots$
 $MP:$
 $(Exists_{MP}) \exists x : Actor \cdot MP(x)$
 $(Unique_{MP}) \forall x, x' : Actor \cdot MP(x) \wedge MP(x') \Rightarrow x = x'$
 $(Distinct_{MP-MS}) \forall x : Actor \cdot MP(x) \Rightarrow \neg MS(x)$
 $(Distinct_{MP-DD}) \forall x : Actor \cdot MP(x) \Rightarrow \neg DD(x)$
 $(Distinct_{MP-MI}) \forall x : Actor \cdot MP(x) \Rightarrow \neg MI(x)$
similarly for all other element and relation predicates

Fig. 3. The *MAVO* predicates and constraints for \mathbf{G} .

element in \mathbf{G} and a binary predicate for each relation instance between the elements in \mathbf{G} . Then, we add constraints to ensure that the only first order structure that satisfies the resulting theory is \mathbf{G} itself:

$$FO(\mathbf{B}_G) = \langle \Sigma \cup \Sigma_G, \Phi \cup \Phi_G \rangle \text{ (see Def. 2)}$$

where Σ_G and Φ_G are model \mathbf{G} -specific predicates and constraints, defined in Fig. 3. We refer to Σ_G and Φ_G as the *MAVO predicates* and *constraints*, respectively.

The FO structures that satisfy $FO(\mathbf{B}_G)$ are those i^* models that satisfy the constraint set Φ_G in Fig. 3. Assume that K is one such i^* model. The *MAVO* constraint *Complete* ensures that K contains no more elements or relation instances than \mathbf{G} . Now consider the actor *MP* in \mathbf{G} . $Exists_{MP}$ says that K contains at least one actor called *MP*, $Unique_{MP}$ – that it contains no more than one actor called *MP*, and the clauses $Distinct_{MP-*}$ – that the actor called *MP* is different from all the other actors. Similar *MAVO* constraints are given for all other elements and relation instances in \mathbf{G} . These constraints ensure that $FO(\mathbf{B}_G)$ has exactly one concretization and thus $K = \mathbf{G}$.

(3) We construct $FO(\mathbf{B})$ from $FO(\mathbf{B}_G)$ by removing constraints corresponding to the annotations in \mathbf{B} . This constraint relaxation allows more concretizations and thus represents increasing uncertainty. If \mathbf{B} were incomplete, we could express this fact by removing the *Complete* clause from Φ_G , thereby allowing concretizations to be those i^* models that extend \mathbf{G} . Similarly, expressing the effect of the *M*, *S* and *V* annotations for an element E corresponds to relaxing Φ_G by removing $Exists_E$, $Unique_E$ and $Distinct_{E-*}$ clauses, respectively. E.g., removing the $Distinct_{DD-*}$ clauses expresses the *V* annotation on the actor *DD* (i.e., it may or may not be distinct from another actor).

IV. ANALYSIS OF GOAL MODELS WITH MAVO UNCERTAINTY

In this section, we describe our key contribution, the combination of goal model analysis with *MAVO* uncertainty annotations, “lifting” goal model analysis from conventional to uncertain goal models.

A. Correctness Condition for MAVO Goal Model Analysis

A *MAVO* goal model represents a set of goal models (i.e., its concretizations), whereas goal model analysis applies only to a single goal model. Thus, correct analysis results over a *MAVO* goal model should aggregate the analysis results over all of its concretizations:

Definition 3 (Correct Labeling): Let M be a *MAVO* goal model. M is *correctly labelled* iff for all intentions I in M , label v is assigned to I iff $\exists m \in [M], i \in I_m \cdot v(i)$, where I_m is the set of intentional elements mapped to I in m .

Thus, a label gets applied to an intention when there is a concretization that has that label on the intention. Model \mathbf{C} in Fig. 4 shows a correct analysis labeling of our example model from Fig. 2. Here we use a shorthand notation for sets of possible labels, e.g., $\checkmark \times \mathbb{N}$ for $\{FS, FD, N\}$ in PiM .

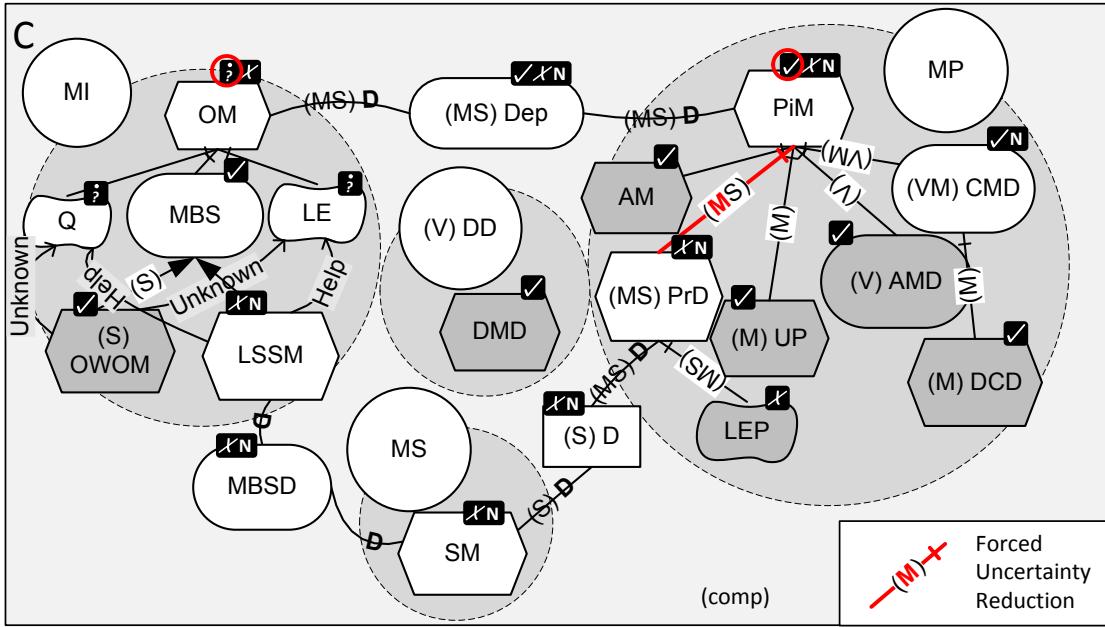


Fig. 4. Analysis of an alternative considering model uncertainty showing forced uncertainty reductions (result of **Q1**).

Thus, the *FS* label in PiM means that there exists at least one concretization in which at least one concrete intention mapped to PiM has label *FS*. An example of such a concretization is one in which PiM has the two sub-intentions **AM** and **AMD** with labels *FS* (they are initial labels indicated by the shading) and the remaining sub-intentions removed (possible because they are M-annotated). In this concretization, PiM gets the propagated label *FS*.

B. Constructing the MAVO Goal Analysis Encoding

In Sec. III-B, we described construction of an FOL encoding $FO(M)$ for every MAVO model M , whose satisfying instances are exactly the concretizations of M (see Fig. 3 for an example). We can incorporate goal analysis into this formalization by adding constraints that express the initial conditions and propagation rules as described in [11]. We define this extended FOL encoding as follows:

Definition 4 (Extended Encoding): Let M be an i^* MAVO model, $FO(M) = \langle \Sigma \cup \Sigma_M, \Phi \cup \Phi_M \rangle$ be its FOL encoding and $\text{Intention} \in \Sigma$ its sort for intentions. The *extended encoding* $FO^e(M)$ is a tuple $FO^e(M) = \langle \Sigma \cup \Sigma_M \cup \Sigma_{label}, \Phi \cup \Phi_M \cup \Phi_l \cup \Phi_p \rangle$, where $\Sigma_{label} = \{\text{FS}(\text{Intention}), \dots, \text{N}(\text{Intention})\}$ are unary predicates representing the possible analysis labels on intentions, Φ_l is the set of constraints encoding initial analysis values and Φ_p is the set of constraints encoding the analysis propagation rules. We define $\Phi_M^e = \Phi \cup \Phi_M \cup \Phi_l \cup \Phi_p$.

The above definition ensures that each instance of $FO^e(M)$ is a concretization of M with the analysis labeling that would result from performing goal analysis on it with the initial labeling defined by Φ_l . Thus, it guarantees that M satisfies the Correct Labeling condition (Def. 3) and we can use $FO^e(M)$ to reason about the analysis results on the concretizations of M . Specifically, we use it to answer the analysis questions posed over our example model in Sec. II.

Initial Analysis Labels (Φ_l). Goal model analysis starts with a set of initial labels representing the analysis question (e.g., the *FS* label applied to *OWOM* and *FD* applied to *LSSM*, i.e., the shaded intentions in model **A** in Fig. 1). On the *MAVO* model level, we interpret these labels as follows: if an intention i in a *MAVO* goal model is given an initial label, then *all* concrete intentions mapped to i must also have this label. A set of constraints Φ_l expresses this condition. For example, to say that the initial *FS* label is applied on *OWOM*, ($FS(OWOM)$), we add the constraint $\forall i : Intention \cdot OWOM(i) \Rightarrow FS(i)$ to Φ_l .

We enforce the constraint that all leaf intentions (intentions with no incoming links) which are not explicitly assigned an initial label must be assigned an explicit N (none) label. Similarly, we enforce a constraint that each intention in the concrete model must have only one analysis label. These initial constraints are added to Φ_l .

Propagation Constraints (Φ_p). Φ_p consists of the FOL encoding of the propagation rules summarized in Sec. III-A and described in [8]. For example, if an *FS*-labelled task is connected to a goal by a *Make* contribution link, the goal should also get label *FS*. This rule is encoded in the original goal model analysis using the predicate $\text{FS}(i_s) \Rightarrow \text{FS}(i_d)$. In order to encode it using *MAVO*, we add the constraint $\forall t : \text{Task}, g : \text{Goal} \cdot (\text{Make}(t, g) \wedge \text{FS}(t)) \Rightarrow \text{FS}(g)$, where *Task* and *Goal* are the sorts for *tasks* and *goals*, respectively, for the *MAVO* model being encoded. We call the set of all such propagation constraints Φ_p . Similar constraints are added for the propagation of other labels on other element types through other links (see the third column of Table II for more examples). The full set of propagation constraints in our encoding covers all combinations of element types, relationships, and intentions.

C. Determining Possible Analysis Labels

We can use our encoding to find analysis results given a particular alternative (set of initial labels) and model uncertainty, answering **Q1**. By Def. 3, a label can hold for an intention in a *MAVO* model iff there exists a concretization containing a corresponding concrete intention with that label. The following proposition shows we can use the extended encoding $FO^e(M)$ to check this condition.

Proposition 1 (Analysis Labeling): Let M be a *MAVO* i* model with an analysis label set assigned to each intention. M is *correctly labelled according to Def. 3* iff for all intentions I in M , label v is assigned to I iff $\Phi_M^e \cup (\exists i : \text{Intention} \cdot I(i) \wedge v(i))$ is satisfiable.

For each possible labeling of intention I with label v , this checks whether the labeling is consistent with Φ_M^e . If so, there must exist a concretization with this labeling. Checking this for all labellings ensures that condition in Sec. 3 is met. For example, for model **C**, to check whether *PiM* should have label *FS*, we ask whether $\Phi_C^e \cup (\exists i : \text{Intention} \cdot \text{PiM}(i) \wedge \text{FS}(i))$ is satisfiable. It is satisfiable since there is a concretization (see Sec. 3) in which *PiM* gets the propagated label *FS*. Performing this check for every possible combination of model element and analysis label allows us to find results across the entire model, as we do in model **C**.

V. REASONING METHODOLOGY

In our motivating scenario, we raised five analysis questions (Table I). We have shown how to answer **Q1**, finding possible analysis labels given uncertainty, in Sec. IV-C. In this section, we describe how the combination of goal model analysis and *MAVO* uncertainty can be used to answer **Q2-Q5**, providing an analysis methodology guiding users in asking the questions and interpreting the answers. We summarize our methodology in Fig. 5. Steps which require manual judgment are indicated by the presence of an actor.

Answering Q2: Making Choices. Given the set of analysis results over the uncertain model, users can select a subset of the results that they consider acceptable. We call this subset *choices*. In our example, the fully satisfied (✓) value is the most desired for *PiM* in Model **C**, while unknown (?) value is the most desired for *OM* (circled in model **C**).

Answering Q3: Checking Simultaneous Achievability of Choices. Once analysis choices have been selected, we can use the extended encoding to check if such choices are simultaneously achievable in one or more concretizations. This is done by encoding the desired label choices as constraints to require that concretizations simultaneously achieve these labels. For example, for model **C**, our choices consist of *U* for *OM* and *FS* for *PiM* and so we define the following constraints: (1) $\forall i : \text{Intention} \cdot \text{OM}(i) \Rightarrow \text{U}(i)$ and (2) $\forall i : \text{Intention} \cdot \text{PiM}(i) \Rightarrow \text{FS}(i)$. We call the set of all such constraints Φ_c and use it to check the existence of a concretization satisfying these choices.

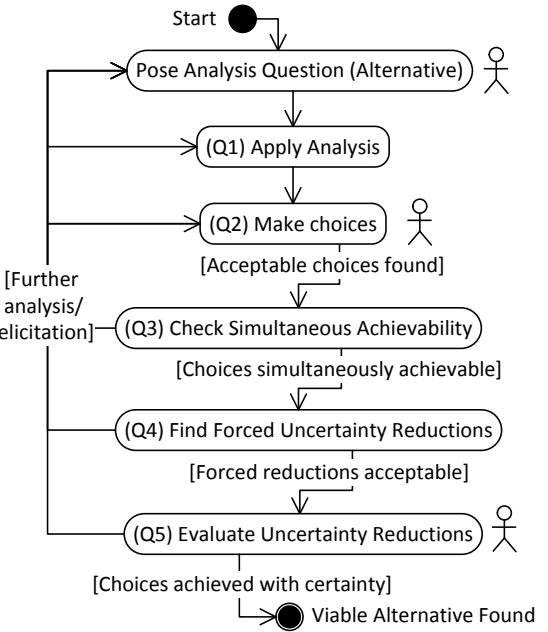


Fig. 5. Goal model analysis methodology with actors indicating manual tasks.

Proposition 2 (Choice Achievability Check): Given a *MAVO* model M and a constraint set Φ_c encoding a set of choices, this set of choices is simultaneously achievable iff $\Phi_M^e \cup \Phi_c$ is satisfiable.

For model **C**, checking the constraints $\Phi_C^e \cup \Phi_c$ shows that our label choices (*U* for *OM* and *FS* for *PiM*) are simultaneously achievable.

Answering Q4: Finding Forced Uncertainty Reductions. Once the users check their choices for simultaneous achievability, we can use the approach defined in [21] to determine which uncertainties are *forced*, i.e., those which must be resolved in all concretizations in order to achieve analysis choices. In our example, the forced uncertainty reductions are those related to *PrD*. In this case, the decomposition link must be removed for our choices to be achieved. We indicate this in model **C** by making the link and the corresponding *M* uncertainty annotation red and bold. Such information allows users to target elicitation on a subset of uncertainties in the model. In our example, users judge this forced resolution to be acceptable, and continue with the analysis process.

Answering Q5: Evaluating Uncertainty Reductions. As an outcome of checking for simultaneous achievability in **Q3**, we produce a concrete model with uncertainties resolved such that choice values are achieved. Model **D** in Fig. 6 is a particular concretization for our example. It shows the meeting scheduler example after a set of uncertainty reductions (in bold) which allow achievement of the chosen analysis results. For example, the *M*-annotated task *PrD* and the uncertainty concerning *UP* have been removed when compared to model **C**.

In our example, users judge some, but not all, of the suggested uncertainty reductions in **D** to be consistent with the domain. For example, it is deemed acceptable that the *MP*

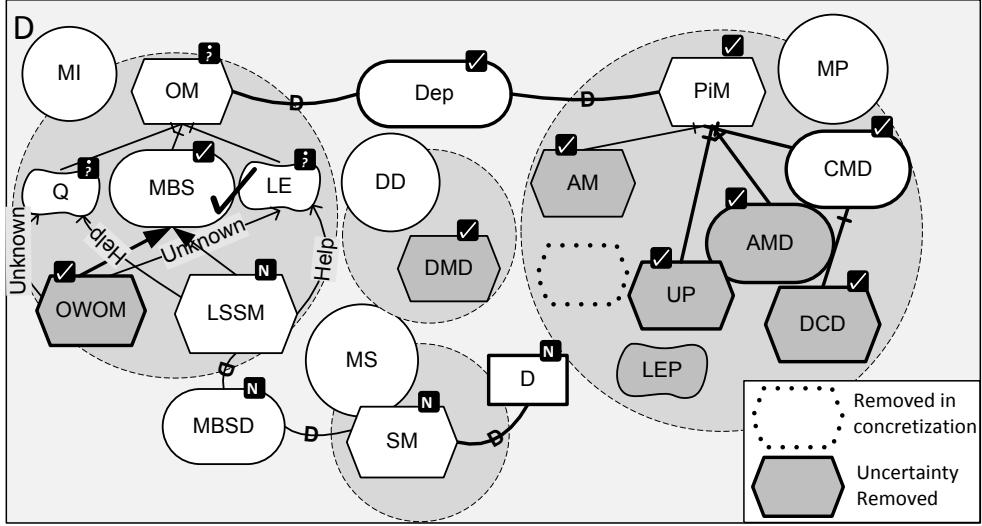


Fig. 6. One possible concretization achieving analysis choices in Model C in Fig. 4 (result of Q3).

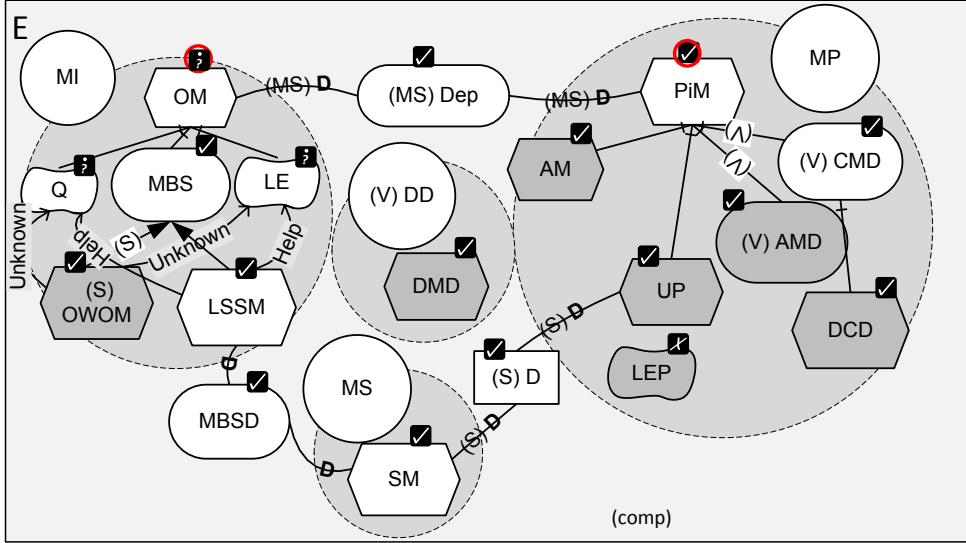


Fig. 7. Chosen and forced uncertainty reductions achieving analysis choices (results of Q5, reapplying Q1).

actor will not perform the task PrD (removed in D); instead, she can keep track of participant details using the UP task. In another example, it is deemed unacceptable for the dependency link from resource D to be removed from the model altogether. This link should now depend on UP instead of the removed PrD. Further uncertainties are accepted or rejected and the model is updated to reflect these decisions (uncertainties are removed, model elements are changed, analysis (Q1) is reapplied).

When iterating over analysis results, different constraints can be added in order to determine if concretizations satisfying them exist, producing one such concretization. For example, when finding concretizations for model C, we may want to specify that actors DD and MP cannot be merged together, expressed by the property Ψ . Checking $\Phi_C^e \cup \Phi_C \cup \{\Psi\}$ produces a concretization in which Ψ holds.

An iterative process of finding concrete models, constraining the problem and accepting or rejecting suggested uncertainty reductions will lead, in the optimal case, to a model in which choices are achieved with certainty and a viable set of alternative requirements is found. Otherwise, the model may not contain viable alternatives, prompting further elicitation. Alternative outcomes are discussed in Sec. VI.

In our running example, iterative analysis directed us to reduce targeted uncertainty in the model and the domain such that we could achieve our analysis choices with certainty. We show the final model resulting from this process, model E, in Fig. 7. In the model without uncertainty, A, we rejected the alternative that included OWOM. Yet considering model uncertainty, as we did in model C, made this alternative viable. We have determined the viability of this alternative without the need to remove all model uncertainty. Applying

this methodology iteratively can enable exploring the viability of other alternatives in the model (e.g., satisfying LMSM), allowing for a final selection between alternative requirements.

VI. EXPERIENCE AND DISCUSSION

Implementation. We have implemented the automatable questions in our methodology (**Q1**, **Q3**, **Q4**), building on top of Model Management Tool Framework (MMTF) [17]. Our implementation automatically converts an i^* model with uncertainty annotation $\langle \Sigma_M, \Phi_M \rangle$, together with the i^* metamodel $\langle \Sigma, \Phi \rangle$, analysis constraints $(\Phi_M \cup \Phi_p)$, and initial labels (Φ_l) , into their corresponding FOL encoding.

Our implementation uses an SMT solver, specifically, Z3 version 4.3.1². It answers **Q1** by iteratively calling the solver, checking each possible label for each intention. We answer **Q3** by adding the user choices to the encoding (Φ_c) and checking for satisfiability. The result is a suggested concrete model. **Q4** is answered using the implementation outlined in [21], integrated into the MMTF Framework.

Experience. Answering questions **Q1**, **Q3**, and **Q4** for the illustrated alternative in the Meeting Scheduler example took 20.72, 0.18, and 7.31 seconds, respectively, using a PC with Intel Core i7-2600 3.40 GHz x 4 cores and 8 GB RAM. We have applied our approach to three additional alternatives for the Meeting Scheduler and to a larger example, Inflo. A description of these models, analysis results, SMT encodings, and running times can be found online³.

Alternative Analysis Paths. Our scheduler example described a successful analysis path through our methodology, leading to a viable analysis alternative with certainty.

However, there may be less successful outcomes which correspond to back-edges in Fig. 5. We claim that even those scenarios that do not directly lead to a viable alternative can help eliminate alternative sets of requirements, reduce uncertainty, and refine the model.

In one scenario, the set of possible analysis values may not yield viable choices, (i.e., **Q2** is answered negatively). For example, if the analysis in model **D** in Fig. 6 yielded only negative values (*PD* or *FD*) for both PiM and OM, users could evaluate further alternatives in the model (e.g., LSSM), or perform further elicitation to discover other requirement alternatives.

In another scenario, if users are able to find sufficient choice values over uncertain results but these values are not simultaneously achievable (**Q3** is answered negatively), users must revise their choices, or evaluate further alternatives. Similarly, if critical uncertainty reductions (**Q4**) are not consistent with the domain, options include relaxing choice values, finding an alternative set of choice values making different trade-offs, or evaluating further alternatives.

If the reductions in uncertainty evaluated as part of **Q5** are not consistent with domain knowledge, users can articulate the reasons for such inconsistencies and either revise the model

and *MAVO* uncertainties to reflect this understanding, or add more specific constraints to the SMT encoding (see Sec. V for an example).

Backward Reasoning. In this paper, we have illustrated our technique using *forward analysis*, i.e., placing labels on leaf elements and asking “what if?” questions. It is also possible to conduct *backward analysis* by placing analysis labels on root elements and asking if it is possible to achieve them and if so, how? [8]. Our current implementation can support backward analysis by removing the constraint which assigns the N label to all leaf intentions not explicitly assigned an initial label. In this case, the set of possible analysis labels presented when answering **Q1** includes possible labels which satisfy backward constraints and labels made possible via uncertainty. Future work will focus on making the sets of possible labels comprehensible to the user, both in forward and backward analysis, e.g., understanding how they can be used to visualize consistent solutions over the whole model.

VII. RELATED WORK

Uncertainty in RE. Several approaches consider uncertainty in requirements, often as part of an overall strategy for managing uncertainty in software development (e.g, [12], [16]). The *MAVO* approach adopted in this work takes a different view on uncertainty in RE, allowing users to capture uncertainty on top of their preferred RE modeling language [20].

Much of the investigation of uncertainty in RE concerns adaptive systems. Such systems aim to respond to run-time uncertainty by specifying functional adaptations as part of RE (see [22] for overview). *MAVO* is aimed to represent uncertainty in the *content*, or structure, of requirements models arising as part of elicitation and design, ideally resolved as part of the software development process, and does not explicitly handle run-time or environmental uncertainty.

Uncertainties in software development are often considered as part of risk management, (e.g, [13]). Although certain risk factors (e.g., an unknown budget) may motivate the presence of model uncertainty, *MAVO* is not explicitly intended to capture risks.

Partial Modeling. May uncertainty in *MAVO* is related to various modal extensions to *behavioural* modeling formalisms. For example, Modal Transition Systems (MTSs) [14] allow introduction of uncertainty about transitions on a given event, whereas Disjunctive Modal Transition Systems (DMTSs) [15] add a constraint that at least one of the possible transitions must be taken in the refinement. Concretizations of these models are Labelled Transition Systems (LTSs). MTSs and DMTSs have been used to capture some forms of uncertainty in early design models [23]. The *MAVO* approach allows specification of more uncertainty types, although it is applicable only to *structural* models.

Goal Model Analysis. Several goal model analysis procedures have been proposed, facilitating decision-making in RE (see [10] for a survey). In this paper, we have focused on qualitative satisfaction analysis. Quantitative approaches (e.g., [7]) are

²<http://z3.codeplex.com/>

³<http://www.cs.utoronto.ca/~jenhork/AnalyzingUncertainGM>

appropriate for later, detailed requirements analysis, where reliable metrics can be found, and are less applicable to probabilistic, early uncertainties. Other approaches consider uncertainty in goal models through varying contexts (e.g., [1]) or by analyzing risk (e.g., [2]). These approaches capture uncertainty inherent in the domain, for example, a change in user context or failure of a component, while our approach focuses on evolving uncertainty arising as part of requirements elicitation, ideally resolved before implementation.

Further approaches have applied SAT solving to goal models, using interpolation to automatically refine operational goals [5]. This technique is not aimed for decision-making or trade-off analysis but to automatically find goal operationalizations which satisfy safety and liveness goals.

Related work has explored the application of search based (SB) techniques to the problem of requirement selection, finding non-dominated sets of optimal requirements [25]. Yet application of SB techniques requires quantitative utility functions mapping all alternatives to all objectives, which are difficult to elicit with accuracy in early RE. Our approach focuses on qualitative analysis over goal models with uncertainty which are more feasible for early decision making.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have motivated the need for support of early decision-making in the presence of uncertainty. We provide such support via an explicit consideration of uncertainty as part of goal model analysis. We have provided a tool-supported methodology, allowing users to answer five analysis questions using uncertain goal models. The result of this process is a reduction of necessary uncertainties, accepting or rejecting alternative requirements with certainty. This work is the first example application of the *MAVO* uncertainty framework [20] taking semantics of a target language into account (in our case, such semantics has been provided in [11], [8]).

While we chose i^* as a particular goal modeling framework and [11] as a particular analysis algorithm, our approach can be applied more generally to work with other goal modeling frameworks (e.g., NFR [3] or KAOS [4]) or other qualitative goal model analysis approaches (e.g., [3], [7]), especially if such approaches come with their own formal semantics, translatable into FOL. We are currently working to improve the usability of *MAVO* labels (e.g., representing uncertainty over link types [6]).

We plan to further evaluate our technique by applying it to industrial RE case studies. The visual presentation of multiple analysis labels should be evaluated and improved through user studies. This work should be further integrated with goal model visualization techniques, helping users understand why the tooling produces no solution for a model [9]. Finally, we are currently considering the consequences of analysis over incomplete models, in a open world.

ACKNOWLEDGMENTS

This work was supported in part by ERC advanced grant 267856, “Lucretius: Foundations for Software Evolution”, www.lucretius.eu.

REFERENCES

- [1] R. Ali, F. Dalpiaz, and P. Giorgini. A Goal-Based Framework for Contextual Requirements Modeling and Analysis. *Requir. Eng.*, 15(4):439–458, Nov. 2010.
- [2] Y. Asnar, V. Bryl, and P. Giorgini. Using Risk Analysis to Evaluate Design Alternatives. In *Proc. of AOSE’07*, pages 140–155, 2007.
- [3] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.
- [4] A. Dardenne, A. V. Lamsweerde, and S. Fickas. Goal-Directed Requirements Acquisition. *Sci. of Comp. Prof.*, 20(1-2):3–50, 1993.
- [5] R. Degiovanni, D. Alrajehy, N. Aguirre, and S. Uchitely. Automated goal operationalisation based on interpolation and sat solving. In *Proc. of ICSE’14*, 2014.
- [6] M. Famelis and S. Santosa. MAV-Vis: A Notation for Model Uncertainty. In *Proc. of MiSE’13*, pages 7–12, 2013.
- [7] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with Goal Models. In *Proc. of ER’02*, volume 3084 of *LNCS*, pages 167–181, 2002.
- [8] J. Horkoff and E. Yu. Finding Solutions in Goal Models: An Interactive Backward Reasoning Approach. In *Proc. of ER’10*, volume 6412 of *LNCS*, page 59, 2010.
- [9] J. Horkoff and E. Yu. Visualizations to support interactive goal model analysis. In *Proc. of REV’10*, pages 1–10, 2010.
- [10] J. Horkoff and E. Yu. Analyzing Goal Models: Different Approaches and How to Choose among Them. In *Proc. of SAC’11*, pages 675–682, 2011.
- [11] J. Horkoff and E. Yu. Interactive goal model analysis for early requirements engineering. *Requirements Engineering (accepted with minor revisions)*, 2014.
- [12] H. Ibrahim, B. H. Far, A. Eberlein, and Y. Daradkeh. Uncertainty Management in Software Engineering: Past, Present, and Future. In *Proc. of CCECE’09*, 2009.
- [13] S. Islam and S. H. Houmb. Integrating Risk Management Activities into Requirements Engineering. In *Proc. of RCIS’10*, pages 299–310, 2010.
- [14] K. G. Larsen and B. Thomsen. A Modal Process Logic. In *Proc. of LICS’88*, 1988.
- [15] P. Larsen. The Expressive Power of Implicit Specifications. In *Proc. of ICALP’91*, volume 510 of *LNCS*, pages 204–216, 1991.
- [16] J. Noppen, P. van den Broek, and M. Akşit. Software Development with Imperfect Information. *J. Soft Computing*, 12(1):3–28, 2008.
- [17] R. Salay, M. Chechik, S. Easterbrook, Z. Diskin, P. McCormick, S. Nejati, M. Sabetzadeh, and P. Viriyakaktyaporn. An Eclipse-Based Tool Framework for Software Model Management. In *Proc. of OOPSLA Eclipse Wrksp.*, pages 55–59, 2007.
- [18] R. Salay, M. Chechik, and J. Horkoff. Managing Requirements Uncertainty with Partial Models. In *Proc. of RE’12*, pages 1–10, 2012.
- [19] R. Salay, M. Chechik, J. Horkoff, and A. Sandro. Managing requirements uncertainty with partial models. *Requirements Engineering*, 18(2):107–128, 2013.
- [20] R. Salay, M. Famelis, and M. Chechik. Language Independent Refinement Using Partial Modeling. In *Proc. of FASE’12*, volume 7212 of *LNCS*, 2012.
- [21] R. Salay, J. Gorzny, and M. Chechik. Change Propagation Due to Uncertainty Change. In *Proc. of FASE’13*, pages 21–36, 2013.
- [22] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein. Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems. In *Proc. of RE’10*, pages 95–103, 2010.
- [23] S. Uchitely and M. Chechik. Merging Partial Behavioural Models. In *Proc. of SIGSOFT FSE’04*, pages 43–52, 2004.
- [24] E. Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In *Proc. of RE’97*, pages 226–235, 1997.
- [25] Y. Zhang, A. Finkelstein, and M. Harman. Search based requirements optimisation: Existing work and challenges. In *Proc. of REFSQ’08*, pages 88–94, 2008.

Integrating Exception Handling in Goal Models

Antoine Cailliau and Axel van Lamsweerde

ICTEAM – Institute for Information & Communication Technologies, Electronics and Applied Mathematics
Université catholique de Louvain
Louvain-la-Neuve, Belgium
{antoine.cailliau, axel.vanlamsweerde}@uclouvain.be

Abstract—Missing requirements are known to be among the major sources of software failure. Incompleteness often results from poor anticipation of what could go wrong with an over-ideal system. Obstacle analysis is a model-based, goal-anchored form of risk analysis aimed at identifying, assessing and resolving exceptional conditions that may obstruct the behavioral goals of the target system. The obstacle resolution step is obviously crucial as it should result in more adequate and more complete requirements. In contrast with obstacle identification and assessment, however, this step has little support beyond a palette of resolution operators encoding tactics for producing isolated countermeasures to single risks. In particular, there is no single clue to date as to where and how such countermeasures should be integrated within a more robust goal model.

To address this problem, the paper describes a systematic technique for integrating obstacle resolutions as countermeasure goals into goal models. The technique is shown to guarantee progress towards a complete goal model; it preserves the correctness of refinements in the overall model; and keeps the original, ideal model visible to avoid cluttering the latter with a combinatorial blow-up of exceptional cases. To allow for this, the goal specification language is slightly extended in order to capture exceptions to goals separately and distinguish normal situations from exceptional ones. The proposed technique is evaluated on a non-trivial ambulance dispatching system.

Index Terms—*Obstacle analysis, goal modeling, probabilistic goals, risk control, requirements completeness, exception handling, goal-oriented requirements engineering, quantitative reasoning.*

I. INTRODUCTION

Requirements-related errors are commonly recognized to be the most frequent, persistent, expensive and dangerous types of software errors [13]. Among these, missing requirements tend to be the worst. They often arise from a natural inclination to believe that the software and its environment will always behave as expected; no requirements are engineered for cases where this optimistic assumption does not hold. Requirements completeness therefore calls for putting risk analysis at the heart of the RE process [2, 3, 5, 8, 13, 14, 20].

A *risk* is an uncertain factor whose occurrence may result in the loss of satisfaction of some high-level objective [4, 8, 13]. A risk has a probability of occurrence and one or multiple consequences. Each consequence has a severity in degree of loss of satisfaction of the corresponding objective [5, 8]. Risks may cover undesirable situations such as safety hazards [16, 18], security threats [12, 26] or data inaccuracies [14] dependent on the type of objective they negatively impact on.

At requirements engineering time, risks should be identified, assessed in terms of their likelihood and criticality,

and controlled through effective countermeasures [13]. Obstacle analysis has been introduced and used as a model-based, goal-oriented form of risk analysis [2, 6, 14, 21]. An *obstacle* to a goal is a precondition for non-satisfaction of this goal. *Obstacle analysis* consists of (a) *identifying* obstacles from available goals, assumptions and domain properties; (b) *assessing* their likelihood and criticality in terms of severity of their consequences; and (c) *resolving* likely and critical obstacles through countermeasures to be incorporated into the goal model.

To support obstacle analysis, techniques are available for *identifying* obstacles systematically from goals and domain properties [1, 14]. For *obstacle assessment*, likelihoods and criticalities may be determined quantitatively by calculations over obstacle refinement trees and goal refinement trees, respectively; such calculations call for probabilistic extensions to cope with probabilistic goals and obstacles [5, 25]. For *obstacle resolution*, operators encoding risk control tactics were proposed to explore alternative resolutions –such as *avoid obstacle*, *reduce obstacle likelihood*, *mitigate obstacle*, *weaken goal*, *substitute goal*, *restore goal*, or *substitute agent* [14].

The obstacle resolution step is obviously crucial; it directly impacts the adequacy, completeness and robustness of the goal model. However, little support is currently available for this step beyond the above resolution operators for countermeasure exploration. In particular, it is totally unclear where and how selected countermeasures produced by such operators should be integrated in the goal model to increase its completeness and robustness.

To address this problem, the paper describes techniques for integrating obstacle resolutions systematically in the goal refinement graph while propagating the resulting changes wherever required in the model. These techniques guarantee that:

- the model is increasingly robust and complete as resolutions are being integrated;
- the normal system behaviors and those not affected by the obstacles are preserved;
- the correctness of goal refinements in the model is preserved.

A goal model integrating countermeasures to obstacles may need to be restructured so as to keep the goals referring to normal situations separate from the countermeasure goals referring to exceptional situations. There are multiple reasons for this.

- For higher readability and better visibility, the ideal model containing all functional and non-functional goals in normal situations should be kept visible. The

specification of these goals should not be cluttered with items referring to exceptional situations.

- The model structuring and specification should not exhibit any combinatorial blow-up of exceptional cases. Without any structuring mechanism, the integration of multiple countermeasures to multiple risks considered in combination might produce a large number of cases.
- Exceptional situations should be identifiable and integrated *incrementally*. The handling of each single situation should be isolated from the others.
- The traceability of exceptional cases should be supported from requirements to architecture. Keeping exceptions separate from each other and from the goals in normal situations enables traceability from the goal model and its operationalization on the one hand and exception handlers in the architecture on the other hand.
- In case of obstacle tolerance with no countermeasure integrated in the model, a one-to-one mapping should be maintained between (a) the obstacle and countermeasure identified at modeling time, and (b) the corresponding runtime monitor/adaptator mechanisms for dynamic reconfiguration when the obstacle is too frequent [9].

To support such separation between normal and exceptional situations, the paper extends the goal language with semantics-preserving constructs for specifying exceptions and their “handlers”—that is, the countermeasures associated with them. Model transformation operators are then provided for attaching and detaching exceptions to/from associated goals in the goal model.

The paper is organized as follows. Section II introduces some necessary background on modeling goals and their obstacles. Section III motivates our technique on a small example. Section IV more precisely specifies the problem to be solved by our integration approach. Section V describes the conditions and mechanisms for integrating countermeasures in a goal model. Section VI presents the constructs for specifying goals with exceptions together with their semantics. Section VII introduces model transformation operators for attaching and detaching exceptions to/from goals. Section VIII summarizes the evaluation of our proposals on an ambulance dispatching system. Section IX briefly discusses related work.

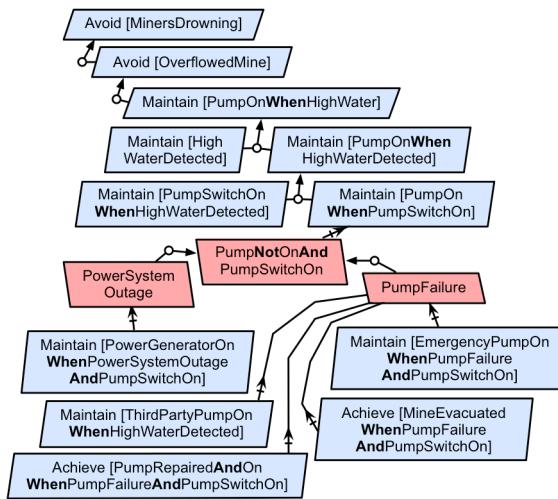


Fig. 1. Partial goal model for a mine pump system

II. BACKGROUND

This section recalls some basics on behavioral goal modeling, obstacle analysis and probabilistic goals while introducing our running example.

Goal-oriented system modeling. A *goal* is a prescriptive statement to be satisfied by cooperating agents forming the considered system. The latter may include devices such as sensors and actuators, people, preexisting software, and the software to be developed. *Domain properties* are descriptive statements about the problem space, e.g., physical laws.

Behavioral goals prescribe maximal sets of desired system behaviors; unlike soft goals they are satisfiable in a clear-cut sense [13]. A *behavior* is a sequence of system state transitions. Linear Temporal Logic (LTL) may be used to specify behavioral goals formally and enable formal analyses [13, 22]. The goals then have the general form:

$$C \Rightarrow \Theta T,$$

where Θ represents an LTL operator such as: \circ (in the next state), \diamond (sometimes in the future), $\diamond_{\leq d}$ (sometimes in the future before deadline d), \square (always in the future), $\square_{\leq d}$ (always in the future up to deadline d), \mathcal{W} (always in the future unless), \mathcal{U} (always in the future until), and where $P \Rightarrow Q$ denotes $\square(P \rightarrow Q)$. The following standard logical connectives are used: \wedge (and), \vee (or), \neg (not), \rightarrow (implies), \leftrightarrow (equivalent).

Among behavioral goals, *Achieve* goals follow the specification pattern “if C then sooner-or-later T ”, that is, $C \Rightarrow \diamond T$, where C and T denote a current and target condition, respectively. *Maintain* goals follow the specification pattern “if C then always G ”, that is, $C \Rightarrow \square G$ where G denotes a good condition. *Avoid* goals follow a similar pattern “if C then never B ”, that is, $C \Rightarrow \square \neg B$ where B denotes a bad condition.

A *goal model* is an AND/OR graph showing how goals contribute positively or negatively to each other. Parent goals are obtained by abstraction, e.g., through *why* questions. Subgoals are obtained by refinement, e.g., through *how* questions. In a goal refinement graph, leaf goals are *requirements* or *assumptions* dependent on whether they are assigned to single software-to-be or environment agents, respectively.

Fig. 1 shows a partial goal model for a mine pump system [11, 13, 15]. Refinement patterns help building such a model through common goal decomposition tactics such as *Milestone-Driven*, *Case-Driven*, *Guard-Introduction*, *Divide-And-Conquer*, *Uncontrollability-Driven*, etc. [7, 13]. For example, consider the following goal in Fig. 1:

Goal Maintain [PumpOnWhenHighWater]

FormalSpec $\forall p:\text{Pump}, s:\text{Sump}$
 $s.\text{WaterLevel} = \text{"High"} \wedge \text{PumpInSump}(p, s)$
 $\Rightarrow p.\text{Motor} = \text{"On"}$

Applying the *Unmonitorability-driven* refinement pattern to this goal yields, after instantiation, the following refinement where the antecedent of the second subgoal becomes monitorable by the software pump controller:

Goal Maintain [HighWaterDetected]

FormalSpec $\forall p:\text{Pump}, s:\text{Sump}, c:\text{PumpController}$
 $s.\text{WaterLevel} = \text{"High"} \wedge \text{PumpInSump}(p, s) \wedge \text{CtrlPump}(c, p)$
 $\Rightarrow c.\text{HighWaterSignal} = \text{"On"}$

Goal Maintain [PumpOnWhenHighWaterDetected]

FormalSpec $\forall p:\text{Pump}, c:\text{PumpController}$
 $c.\text{HighWaterSignal} = \text{"On"} \wedge \text{CtrlPump}(c, p)$
 $\Rightarrow p.\text{Motor} = \text{"On"}$

Refinement patterns produce goal refinements guaranteed to be complete, consistent and minimal. A refinement is *complete* when the subgoals SG_i , possibly with domain properties in Dom , are sufficient for satisfying the parent goal PG :

$$\{SG_1, \dots, SG_n, Dom\} \models PG \quad (\text{complete refinement})$$

A refinement is *consistent* if:

$$\{SG_1, \dots, SG_n, Dom\} \neq \text{false} \quad (\text{consistent refinement})$$

A refinement is *minimal* if all subgoals are needed for satisfaction of the parent goal:

$$\text{for all } 1 < i < n: \{SG_1, \dots, SG_{i-1}, SG_{i+1}, \dots, SG_n, Dom\} \neq PG$$

The two lower AND-refinements in the upper part of Fig. 1 are complete, consistent and minimal.

Obstacle analysis. An *obstacle* to a goal is a satisfiable precondition for not satisfying this goal [14]:

$$\{O, Dom\} \models \neg G \quad (\text{obstruction})$$

$$\{O, Dom\} \neq \text{false} \quad (\text{domain consistency})$$

The obstacles to a goal may similarly be organized as an AND/OR refinement tree. The root obstacle is the negation of the obstructed goal. An AND-refinement captures a combination of subobstacles to satisfy the parent obstacle. An OR-refinement captures alternative ways of satisfying the parent. Each OR-refinement SO_i must entail the parent obstacle PO :

$$\text{for all } i: \{SO_i, Dom\} \models PO \quad (\text{entailment})$$

OR-refinements should be domain-complete and disjoint:

$$\{\neg SO_1, \dots, \neg SO_n\} \models \neg PO \quad (\text{domain completeness})$$

$$\text{for all } i \neq j: \{SO_i, SO_j, Dom\} \neq \text{false} \quad (\text{disjointness})$$

Leaf obstacles are fine-grained obstacles whose satisfiability and likelihood can be more easily estimated by experts.

A variety of formal techniques, obstruction patterns and heuristic rules are available for systematic obstacle identification [1, 13, 14]. For example, consider the following right leaf goal in Fig. 1:

Goal Maintain [PumpOnWhenPumpSwitchOn]

FormalSpec $\forall p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \Rightarrow p.\text{Motor} = \text{"On"}$

Its negation yields the following root obstacle:

Obstacle PumpNotOnAndPumpSwitchOn

FormalSpec $\diamond \exists p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \wedge p.\text{Motor} \neq \text{"On"}$

Consider the following domain property capturing a necessary condition for the target $p.\text{Motor} \neq \text{"On"}$:

$$p.\text{Failure} = \text{true} \Rightarrow p.\text{Motor} \neq \text{"On"}$$

Regressing the root obstacle backwards through this domain property generates the following subobstacle:

Obstacle PumpFailure
FormalSpec $\diamond \exists p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \wedge p.\text{Failure} = \text{true}$

A variety of tactics are available for exploring alternative ways of resolving obstacles to a goal –such as *avoid obstacle*, *reduce obstacle likelihood*, *mitigate obstacle*, *weaken goal*, *substitute goal*, *restore goal*, or *substitute agent* [14]. For example, the obstacle mitigation tactic applied to the preceding obstacle PumpFailure may produce the following countermeasure goal :

Goal Achieve[MineEvacuatedWhenPumpFailureAndPumpSwitchOn]

FormalSpec $\forall m: \text{Miner}, p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \wedge p.\text{Failure} = \text{true} \Rightarrow \diamond_{\leq 10m} m.\text{Position} = \text{"Out"}$

The lower part of Fig. 1 shows two OR-refinements for the obstacle PumpNotOnAndPumpSwitchOn.

Probabilistic goals and obstacles. Behavioral goals may be satisfied only partially [5]. The *probability of satisfaction* for a goal $G: C \Rightarrow \Theta T$ is defined as the ratio between (a) the number of possible behaviors satisfying both the goal antecedent C and consequent ΘT , and (b) the number of possible behaviors satisfying C . The *estimated probability of satisfaction* (EPS) of a goal is the probability of its satisfaction in view of its possible obstructions by obstacles. For goal G , it is denoted by $P(G)$. The conditional probability $P(G|H)$ denotes the probability of satisfaction of G over all behaviors satisfying H . The *required degree of satisfaction* (RDS) of a goal is the minimal probability of satisfaction admissible for this goal; it is prescribed by elicited requirements, existing regulations, standards, etc. For goal G , it is denoted by $RDS(G)$.

The *probability of an obstacle* is defined as the ratio between (a) the number of possible behaviors satisfying the obstacle, and (b) the number of possible behaviors. The probability of a root obstacle is computed by up-propagation through the obstacle refinement tree from estimates for leaf obstacles. The result is then up-propagated in turn through the AND/OR goal graph to determine the EPS of root goals. The severity of the consequences of obstacles to G is then assessed from the difference $P(G) - RDS(G)$ [5].

III. INTEGRATING COUNTERMEASURE GOALS: MOTIVATION

In addition to the PumpFailure obstacle in the previous section, the following other obstacle also results in severe loss of satisfaction of high-level goals:

Obstacle PowerSystemOutage

FormalSpec $\diamond \exists p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \wedge p.\text{PowerLine} = \text{"Off"}$

Resolution tactics might produce, among others, the following countermeasure goals to resolve these obstacles (see Fig. 1):

Achieve [PumpRepairedAndOnWhenPumpFailure

And PumpSwitchOn],

Maintain [EmergencyPumpOnWhenPumpFailure

And PumpSwitchOn],

Maintain [ThirdPartyPumpOnWhenHighWaterDetected],

Achieve [PowerGeneratorOnWhenPowerSystemOutage

And PumpSwitchOn].

The paper aims at providing precise and systematic answers to the following questions.

- Where should these countermeasure goals be integrated into the goal refinement graph? How?
- What parent goals should the countermeasure goals refine? With what other sibling subgoals? Should exceptional conditions such as $p.\text{Failure} = \text{true}$ and $p.\text{PowerLine} = \text{"Off"}$ pollute goal specifications throughout the entire goal model?

The answers to those questions should support the following objectives.

- *Separation of concerns.* The goals referring to normal situations should be distinguished from those handling obstacle occurrences. Such separation may significantly reduce model complexity and keep the ideal, normal model explicit.
- *Compositional.* It should be possible to specify, structure, and analyze normal goals and countermeasures to their obstacles in a compositional way. A robust model

should not exhibit goal specifications or refinements intermixing normal cases and countermeasures. For example, consider a goal G obstructed by obstacles O_1 and O_2 with corresponding countermeasure CG_1 and CG_2 , respectively. A brute-force integration might produce a cluttered specification for the robust version of this goal such as:

$$(\neg O_1 \wedge \neg O_2 \Rightarrow G) \wedge (O_1 \Rightarrow CG_1) \wedge (O_2 \Rightarrow CG_2)$$

The refinement of such a goal specification and, recursively, of its subgoals would thus intermix normal cases and combinations of exceptional cases which may lead, for a large number of obstacles, to a combinatorial blow-up of cases along refinements.

- *No premature decision and freedom of choice.* The modeler should be free to decide, when felt necessary, what are the goals referring to normal situations and what are those corresponding to exceptional situations. In our example, mine evacuation might or might not be part of the normal situation depending on its frequency, criticality, stakeholder wishes, and so forth.

IV. THE COUNTERMEASURE INTEGRATION PROBLEM

Integrating a countermeasure goal to an obstacle in a goal model means: (a) connecting this goal to a parent node in the model; (b) adding other sibling subgoals in the refinement if necessary; (c) propagating the corresponding changes along refinement trees in which the countermeasure goal is involved; and (d) refining this goal if necessary.

More precisely, an integration $Int_{CG,O}(M)$ of countermeasure CG to obstacle O in goal model M maps M to a new model M' so as to satisfy the following desired properties.

1. *Progress.* The application of the integration operator $Int_{CG,O}$ should increase the probability of satisfaction of some root goal G at least, that is, there must be at least one root goal G' in M' corresponding to G in M such that $P_M(G') > P_M(G)$.
2. *Minimal change.* The application of $Int_{CG,O}$ should preserve prescribed behaviors in M that are not affected by O , that is, for any goal G in M such that $\{O, Dom\} \neq \neg G$ and corresponding goal G' in M' , we should have: $G' \models G$.
3. *Correctness preservation.* The correctness of goal refinements in M should be preserved in M' , that is, if all refinements in M are complete, consistent, and minimal then those in M' are complete, consistent, and minimal as well.

V. INTEGRATING COUNTERMEASURE GOALS

This section explains how our integration operator ensures the three preceding properties.

A. Ensuring progress: valid countermeasures

For the *progress* constraint to be met, two conditions must hold on the countermeasure goal to be integrated.

1. *Non-obstruction.* The countermeasure goal CG may no longer be obstructed by the obstacle O it resolves:

$$\{O, Dom\} \neq \neg CG \quad (\text{non-obstruction})$$

2. *Ancestor entailment.* The countermeasure goal CG must guarantee a deidealized version of an ancestor of the leaf goal obstructed by O . To make this further precise we need the following definitions.

A goal G' is a *deidealized version* of goal G if $G \models G'$.

A deidealized version G' of G is *acceptable* if, for every goal refinement with G as a child, there exists an acceptable deidealized version of its siblings and parents such that the corresponding refinement still meets the completeness, consistency and minimality conditions recalled in Section II.

The ancestor entailment condition can now be formulated as follows:

$$\{CG, G_1', \dots, G_n', Dom'\} \models PG' \quad (\text{ancestor-entailment})$$

for some acceptable deidealized version PG' of ancestor PG , where PG is an ancestor of the obstructed goal G and G_1', \dots, G_n' are acceptable deidealized versions of descendants of PG .

A countermeasure goal CG against obstacle O is said to be *valid* if it satisfies the *non-obstruction* and *ancestor-entailment* conditions.

For example, the countermeasure goal *Achieve [MineEvacuatedWhenPumpFailureAndPumpSwitchOn]* is valid; the obstacle *PumpFailure* does not obstruct it, and this goal together with the deidealized goal *Avoid [OverflowedMineWhenNoPumpFailure]* guarantee the satisfaction of the parent goal *Avoid [MinersDrowning]*. In this example, the parent goal is not deidealized.

Obstacle resolution tactics such as *avoid obstacle*, *reduce obstacle likelihood*, *mitigate obstacle*, *weaken goal*, or *substitute goal* [14] can be shown to produce valid countermeasures modulo propagations of their effect through the refinement trees in which they are involved (see Section V.C).

Multiple candidate ancestors might be considered for the *ancestor-entailment* condition. In our example, *Avoid [OverflowedMine]* and *Avoid [MinersDrowning]* are potential candidates with respect to the goal *Achieve [MineEvacuatedWhenPumpFailureAndPumpSwitchOn]*. The nearest candidate to this goal appears preferable for more local model change –that is, the goal *Avoid [OverflowedMine]*.

The *anchor* for a countermeasure goal CG is the lowest ancestor goal PG meeting the *ancestor-entailment* condition. It is the goal through which the countermeasure goal is integrated, as discussed in the next section.

Theorem (Progress). For any valid countermeasure goal CG , the probability of satisfaction of its anchor PG increases:

$$P(PG') > P(PG),$$

where PG' in M' corresponds to PG in M .

This can be proved *ab absurdo*. Assume there is no such increase: $P(PG') \leq P(PG)$. Introducing conditional probabilities, we have:

$$P(PG') = P(O) \times P(PG'|O) + P(\neg O) \times P(PG'|\neg O),$$

$$P(PG) = P(O) \times P(PG|O) + P(\neg O) \times P(PG|\neg O).$$

Given the obstruction of PG , we have: $P(PG|O) = 0$. Therefore,

$$\begin{aligned} P(O) \times P(PG|O) + P(\neg O) \times P(PG'|\neg O) \\ \leq P(\neg O) \times P(PG|\neg O) \end{aligned}$$

Since PG' is a deidealized version of PG , we have:

$$P(PG'|\neg O) \geq P(PG|\neg O).$$

Therefore,

$$P(\neg O) \times P(PG'|\neg O) \geq P(\neg O) \times P(PG|\neg O).$$

As $P(O) \times P(PG|O) \geq 0$, our initial assumption gets contradicted.

B. Ensuring minimal changes: integration schemas

A single valid countermeasure goal ensures progress towards a complete model; its integration in the model should also ensure that the *minimal change* property is met (see Section IV).

Two alternative integration schemas may be used for this, dependent on the obstacle resolution tactic being selected [14]. The first schema removes the obstructed goal; it should be applied when the *substitute goal* or *weaken goal* tactic is used for resolving the obstacle. The second integration schema keeps the obstructed goal in the model; it should be applied when the *avoid obstacle*, *reduce obstacle likelihood*, or *mitigate obstacle* tactic is used.

Removing the obstructed goal. Fig. 2 shows a first integration schema expressed as a model rewriting rule. In this first schema, the refinement of anchor goal AG , containing at least one obstructed goal, is replaced with a new refinement. The latter contains the countermeasure goal CG to leaf obstacle LO together with all non-obstructed children. (An anchor's *obstructed children* are those being directly or indirectly obstructed by LO .) The anchor AG may need to be deidealized; in this case, AG' replaces AG in the new goal model.

This first integration schema has a precondition for use, namely, the countermeasure goal CG and the non-obstructed children are sufficient for satisfying the anchor goal:

$$\{CG, \text{Children}(AG) \setminus \text{ObstructedChildren}\} \models AG'.$$

Otherwise, the new refinement would not be complete.

For example, the goal $\text{Maintain}[\text{ThirdPartyPumpOnWhenHighWaterDetected}]$ is a countermeasure produced through the *goal substitution* tactic [14]. Its anchor goal is $\text{Maintain}[\text{PumpOnWhenHighWater}]$. The following refinement is complete, consistent, and minimal for this anchor goal:

$$\begin{aligned} & \text{Maintain}[\text{PumpOnWhenHighWater}] \\ & \leftarrow \text{Maintain}[\text{HighWaterDetectedWhenHighWater}] \\ & \leftarrow \text{Maintain}[\text{ThirdPartyPumpOnWhenHighWaterDetected}] \end{aligned}$$

We may therefore replace the old refinement with this one.

It is easy to see that this first integration schema meets our *minimal change* property. Non-obstructed goals are composed from non-obstructed goals in the refinements of AG and its descendants, and in the siblings of AG and its ancestors. The former are kept as is whereas the latter might need to be deidealized through change propagation (see Section V.C). All these goals thus satisfy $G' \models G$.

Keeping the obstructed goal. Fig. 3 shows a second integration schema. In this schema, a new refinement is introduced; it includes a modified version of the obstructed anchor and the countermeasure goal. The obstructed anchor is deidealized for removing the obstruction. The negation of the leaf obstacle is added to form a decomposition by cases.

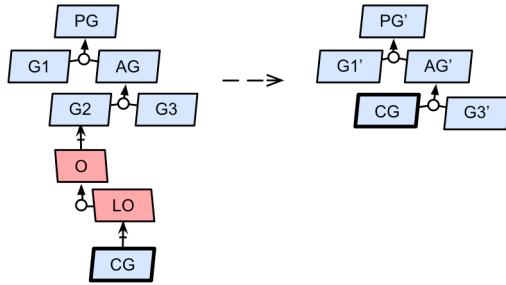


Fig. 2. Integration schema with obstructed goal being removed

This second integration schema has a precondition for use as well; the countermeasure goal must be sufficient for satisfying the anchor goal when the obstacle occurs:

$$\{LO, CG, \text{Dom}\} \models AG'$$

For example, the countermeasure goal $\text{Achieve}[\text{MineEvacuatedWhenPumpFailureAndPumpSwitchOn}]$ entails its anchor $\text{Avoid}[\text{MinersDrowning}]$ when the obstacle occurs. This second rule then produces the following refinement:

$$\text{Avoid}[\text{OverflowedMine}]$$

$$\leftarrow \text{Avoid}[\text{OverflowedMineWhenNoPumpFailure}]$$

$$\leftarrow \text{Achieve}[\text{MineEvacuatedWhenPumpFailureAndPumpSwitchOn}]$$

Note that the anchor goal is not deidealized in this example. The new goal $\text{Avoid}[\text{OverflowedMineWhenNoPumpFailure}]$ is a deidealization of the anchor goal. The negation of the obstacle is added to the goal antecedent:

$$\text{Goal Avoid}[\text{OverflowedMine}]$$

$$\text{FormalSpec } \forall p: \text{Pump}$$

$$p.\text{Failure} = \text{false} \Rightarrow \neg \text{OverflowedMine}$$

This second integration schema can be seen to meet our *minimal change* property as well; the reasoning is similar to the first schema. Only siblings of the anchor goal and its ancestor may need to be deidealized. The other goals were obstructed by the resolved obstacle and are therefore not concerned with the *minimal change* property.

C. Preserving refinement correctness: change propagation

As introduced before, goal deidealizations and countermeasure integrations may require corresponding changes to be propagated along refinement trees in which the countermeasure goal is involved. Such propagations are intended to ensure our third property on integrations, that is, the resulting model must remain complete, consistent, and minimal. This section discusses how deidealizations and change propagations are performed.

Deidealization by strengthening the goal's antecedent. A first way of deidealizing a goal of form $C \Rightarrow \Theta T$ is to add an adequate conjunct to its antecedent:

$$\text{AddConjunct}(C \Rightarrow \Theta T, \Theta EC) = [C \wedge \Theta EC] \Rightarrow \Theta T.$$

For example, the goal $\text{Maintain}[\text{PumpOnWhenPumpSwitchOn}]$ may be deidealized so as to exclude pump failure from pump actuation:

$$\text{Goal Maintain}[\text{PumpOnWhenNoPumpFailureAndPumpSwitchOn}]$$

$$\text{FormalSpec } \forall p: \text{Pump}$$

$$[p.\text{Switch} = \text{"On"} \wedge p.\text{Failure} = \text{false}] \Rightarrow p.\text{Motor} = \text{"On"}$$

Deidealization by weakening the goal's consequent. A second way of deidealizing the goal $C \Rightarrow \Theta T$ is to add an adequate disjunct to its consequent:

$$\text{AddDisjunct}(C \Rightarrow \Theta T, \Theta ED) = C \Rightarrow [\Theta T \vee \Theta ED].$$

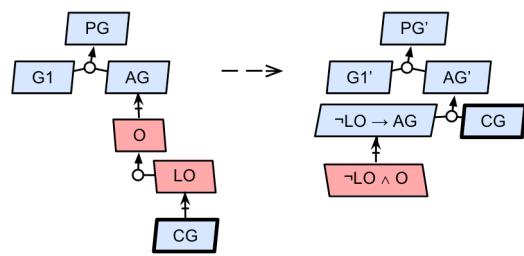


Fig. 3. Integration schema with obstructed goal being kept

For example, the goal `Maintain[PumpOnWhenPumpSwitchOn]` might be deidealized so as to require the pump to be actuated or the emergency pump to be activated:

Goal `Maintain [PumpOnOrEmergencyPumpOnWhenPumpSwitchOn]`
FormalSpec $\forall p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \Rightarrow [p.\text{Motor} = \text{"On"}]$
 $\vee \exists \text{ep: EmergencyPump} \cdot \text{ep.Switch} = \text{"On"}$

Change propagation. When a goal is deidealized, the change must be propagated along the refinement trees in which this goal is involved –both up and down such trees. The *AddConjunction* or *AddDisjunction* operators therefore have to be applied recursively to the other goals up and down refinement links.

For example, the integration of the countermeasure goal `Achieve[MineEvacuatedWhenPumpFailureAndPumpSwitchOn]` requires change propagation to the descendants of the goal `Avoid[OverflowedMine]`. First, this goal is modified as previously shown. Next, the change is down-propagated, leading to an application of *AddConjunction* to the goal `Maintain[PumpOnWhenHighWater]`:

Goal `Maintain [PumpOnWhenNoPumpFailureAndHighWater]`
FormalSpec $\forall p: \text{Pump}, s: \text{Sump}$
 $s.\text{WaterLevel} = \text{"High"} \wedge \text{PumpInSump}(p, s) \wedge p.\text{Failure} = \text{false}$
 $\Rightarrow p.\text{Motor} = \text{"On"}$

The next refinement instantiates the *unmonitorability-driven* refinement pattern. The *AddConjunction* operator is therefore applied to the goal `Maintain[PumpOnWhenHighWaterDetected]`. We obtain:

Goal `Maintain [PumpOnWhenNoPumpFailureAndHighWaterDetected]`
FormalSpec $\forall p: \text{Pump}, c: \text{PumpController}$
 $c.\text{HighWaterSignal} = \text{"On"} \wedge \text{CtrlPump}(c, p) \wedge p.\text{Failure} = \text{false}$
 $\Rightarrow p.\text{Motor} = \text{"On"}$

The next refinement instantiates the *milestone-driven* refinement pattern. The *AddConjunction* operator is therefore applied to both children. We thereby obtain:

Goal `Maintain [PumpSwitchOnWhenNoPumpFailureAndHighWaterDetected]`
FormalSpec $\forall p: \text{Pump}, c: \text{PumpController}$
 $c.\text{HighWaterSignal} = \text{"On"} \wedge \text{CtrlPump}(c, p) \wedge p.\text{Failure} = \text{false}$
 $\Rightarrow p.\text{Switch} = \text{"On"}$

Goal `Maintain [PumpOnWhenNoPumpFailureAndPumpSwitchOn]`
FormalSpec $\forall p: \text{Pump}, c: \text{PumpController}$
 $p.\text{Switch} = \text{"On"} \wedge p.\text{Failure} = \text{false} \Rightarrow p.\text{Motor} = \text{"On"}$

Since these goals are leaf goals, the propagation ends.

Change propagation in the general case proceeds as follows. When *AddConjunction* or *AddDisjunction* is applied to a goal for deidealization, the pattern used for refining (resp. abstracting) it is identified. A *propagation pattern* associated with the refinement/abstraction pattern tells us what goals in the refinement (resp. abstraction) must be modified and how. The process is applied recursively to the subgoals (resp. parents) until leaf goals (resp. root goals) are reached.

For example, if the *case-driven* pattern is used for refining a goal through multiple disjoint cases [13], the application of *AddConjunction* or *AddDisjunction* to a child goal requires the application of the same operator to the parent goal (and vice-versa). If the *milestone-driven* pattern is used for refining a goal through milestone subgoals [13], an application of *AddConjunction* to the first milestone subgoal requires the application of the same operator to the parent goal –but not necessarily to the other subgoals; the corresponding extra

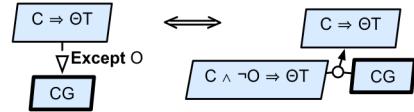


Fig. 4. Semantic equivalent of *Except*

condition is not necessarily relevant to the latter.

For a given refinement pattern and application of *AddConjunction* or *AddDisjunction* to a goal, there might be alternative modifications of the refinement/abstraction structure. Moreover, other mechanisms are required for change propagation to goals not obtained through refinement patterns. Even though pattern-based propagation can be performed semi-automatically, the general problem of automatic change propagation through arbitrary refinement structures remains open.

VI. OBSTACLE RESOLUTION AS EXCEPTION HANDLING

The integration of countermeasure goals through new, explicit refinements in the original model raises several issues.

- The goal graph might undergo significant changes each time a new obstacle is identified.
- Normal situations would be mixed with exceptional ones; it might be hard to distinguish the former from the latter without domain expertise.
- Goal specifications become increasingly more complex.
- As new countermeasures are introduced, the ordered nesting of exceptional cases along refinements may lead to a combinatorial blow-up of special cases.

This section introduces a slight extension of the goal specification language that solves those issues. Dedicated constructs are provided for encapsulating the required modifications while documenting each exceptional case separately.

A. Extending the goal specification language

Except. A first construct links a countermeasure goal to its anchor goal:

Goal AG
FormalSpec $C \Rightarrow \Theta T$
Except O then CG,

where *AG* denotes the anchor goal $C \Rightarrow \Theta T$ of countermeasure goal *CG* to obstacle *O*. Semantically, this implicit specification is fully equivalent to the refinement in Fig. 4.

This construct may be used under the following precondition:

$$\{O, CG, \text{Dom}\} \vDash AG$$

For example, the goal `Avoid[MinersDrowning]` is satisfied in the ideal situation by avoiding mine overflow. Under the exceptional condition of a pump failure, the goal is guaranteed through miners evacuation. We may therefore write:

Goal `Avoid [MinersDrowning]`
Except `PumpFailure`
then `Achieve [MineEvacuatedWhenPumpFailureAndPumpSwitchOn]`

This specification is logically equivalent to the refinement illustrating the second integration schema in Section V.B.

Multiple *Except* annotations may be attached to a single goal to cope with different obstacles; the latter may therefore be introduced incrementally. Compared with the complexity of an equivalent explicit specification, the complexity of an implicit goal specification with multiple *Except* annotations

remains linear in the number of exceptions. The specification of the ideal goal remains unchanged. Moreover, multiple annotations sharing the same countermeasure goal may be factored out to simplify the model.

Provided. A second construct specifies an extra conjunct on the antecedent of an ideal goal G to produce a countermeasure:

Goal G

FormalSpec $C \Rightarrow \Theta T$
Provided EC ,

where EC denotes an extra conjunct to be added to G 's antecedent for resolving the considered obstacle. Semantically, this implicit specification is equivalent to:

Goal G

FormalSpec $[C \wedge EC] \Rightarrow \Theta T$.

The **Provided** construct is typically used for deidealizing goals. For example, the deidealization of the goal **Maintain** [**PumpOnWhenPumpSwitchOn**] may be specified by highlighting the normal situation as follows:

Goal **Maintain** [**PumpOnWhenPumpSwitchOn**]

FormalSpec $\forall p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \Rightarrow p.\text{Motor} = \text{"On"}$
Provided $p.\text{Failure} \neq \text{false}$.

It often proves convenient to write **ProvidedNot** EC instead of **Provided** $\neg EC$.

RelaxedTo. Symmetrically to **Provided**, this construct specifies an extra disjunct on the consequent of an ideal goal G to produce a countermeasure:

Goal G

FormalSpec $C \Rightarrow \Theta T$
RelaxedTo ED ,

where ED denotes an extra disjunct to be added to G 's consequent for resolving the considered obstacle. Semantically, this goal is equivalent to:

Goal G

FormalSpec $C \Rightarrow [\Theta T \vee ED]$.

This construct is useful for deidealizing goals as well. For example, another deidealization of the same goal **Maintain** [**PumpOnWhenPumpSwitchOn**] might be specified by highlighting the normal situation as follows:

Goal **Maintain** [**PumpOnWhenPumpSwitchOn**]

FormalSpec $\forall p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \Rightarrow p.\text{Motor} = \text{"On"}$
RelaxedTo $\exists ep: \text{EmergencyPump} \cdot ep.\text{Switch} = \text{"On"}$.

Multiple **Provided** and **RelaxedTo** annotations may be attached to a single goal to introduce multiple countermeasures.

Replaces. This construct appears useful for tracing previous versions of a goal:

Goal G'

Replaces G

Such traceability helps readers understand the rationale behind the final goal, e.g.,

Goal **Maintain** [**ThirdPartyPumpOnWhenPumpSwitchOn**]
Replaces **Maintain** [**PumpOnWhenPumpSwitchOn**].

B. Exception diagrams

Textual goal specifications with **Except** and **Replaces** annotations may be graphically represented in an exception diagram. Fig. 5 shows a portion of such a diagram for the goal **Maintain** [**ThirdPartyPumpOnWhenPumpSwitchOn**]. This diagram captures that (a) when the obstacle **PumpFailure** occurs the

countermeasure goal **Achieve** [**PumpRepairedAndOnWhenPumpFailureAndPumpSwitchOn**] will guarantee this goal, and (b) this goal replaces **Maintain** [**PumpOnWhenPumpSwitchOn**]. Note that the **Except** annotation has been propagated to the replacing

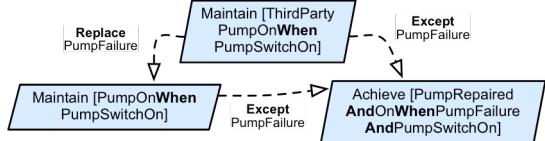


Fig. 5. Exception diagram

goal.

VII. MODEL REFACTORING FOR ATTACHING OR DETACHING GOAL EXCEPTIONS

In practice, the analyst should decide at some point whether a countermeasure goal refers to an exceptional situation or to a normal one to be considered in the ideal model. Such a decision might depend on various factors such as the frequency of the resolved obstacle, the criticality of the obstructed goal, domain-specific culture, stakeholders wishes, and so forth. To make the decision flexible and easily reversible, this section presents model refactoring operators for attaching or detaching the annotations introduced in Section VI to/from a goal model.

Three operators are available for transforming an annotated model portion into a standard one.

Detach-Except, applied to an annotated goal, produces a new model where the goal is no longer annotated with a specific **Except** clause. The operator introduces a new refinement with two children: the countermeasure goal and a deidealization of the original goal (see Fig. 4 from left to right). The children of the original goal are then children of the deidealized goal. Back to an earlier example, the operator takes the model fragment in Fig. 6a to produce the model fragment in Fig. 6b.

Detach-Provided, applied to an annotated goal, produces a new model where a specific **Provided** annotation is “compiled” into its equivalent formal specification. For example, after application of this operator the goal **Maintain** [**PumpOnWhenPumpSwitchOn**] is specified without its **Provided** annotation as follows:

Goal **Maintain** [**PumpOnWhenPumpSwitchOn**]

FormalSpec $\forall p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \wedge p.\text{Failure} = \text{false} \Rightarrow p.\text{Motor} = \text{"On"}$

Detach-RelaxedTo, applied to an annotated goal, produces a new model where a specific **RelaxedTo** annotation is removed. Back to an earlier example, the application of this operator to the goal **Maintain** [**PumpOnWhenPumpSwitchOn**] yields the following goal specification:

Goal **Maintain** [**PumpOnWhenPumpSwitchOn**]

FormalSpec $\forall p: \text{Pump}$
 $p.\text{Switch} = \text{"On"} \Rightarrow [p.\text{Motor} = \text{"On"}$
 $\vee (\exists ep: \text{EmergencyPump}) ep.\text{Switch} = \text{"On"}$]

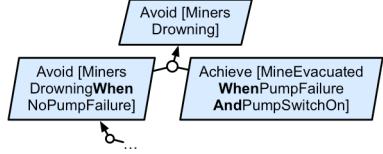
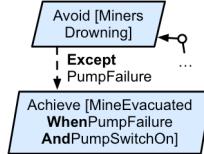


Fig. 6. Implicit and explicit countermeasure integration

Similarly, three operators are available for transforming a standard model portion into an annotated one –namely, *Attach-Except*, *Attach-Provided* and *Attach-RelaxedTo*. These operators are the reverse of the *Detach* ones.

VIII. EVALUATION

The techniques presented in this paper were applied¹ to a benchmark commonly used for evaluating obstacle analysis techniques [1, 14, 15]. The goal and obstacle models used for the London Ambulance System (LAS) are based on [14].

The goal model contains 42 goals, 19 refinements instantiating a variety of refinement patterns, and 8 agents. The obstacle model contains 71 obstacles and 30 countermeasure goals. The full models can be found in [14, 15]. Only portions of the goal model are considered here.

The top goal in this model is Achieve [IncidentResolved]. The *milestone-driven* refinement pattern produces three subgoals: Achieve [IncidentReported], Achieve [AmbulanceOnSceneWhen IncidentReported] and Achieve [IncidentResolvedWhenAmbulance OnScene]. At a lower level of refinement, the goal Achieve [AmbulanceMobilizedWhenAllocated] states that allocated ambulances shall be mobilized within 3 minutes. This goal is refined using a *case-driven* pattern into Achieve [Ambulance MobilizedAtStationWhenAllocated] and Achieve [AmbulanceMobilized OnRoadWhenAllocated]. These goals are in turn refined until they are assignable to single agents.

Obstacles to leaf goals were then generated and refined. Here is a sample of obstacle refinements in textual format:

```
MobilizationOrderPrintedAndAmbulanceNotMobilized
  ← AmbulanceNoLongerAtStation
  ← AmbulanceNoLongerAvailable
  ← MobilizationOrderIgnored
  ← MobilizedToWrongDestination
  ← MobilizationOrderTakenByOtherAmbulance
```

Countermeasures goals were explored using available resolution tactics [14]. For example, here are countermeasure goals for two leaf obstacles:

```
MobilizationOrderTakenByOtherAmbulance
  ← Achieve [MobilizationByOtherAmbulanceKnown]
  ← Avoid [MobilizationWithoutOrder]
MDT-MobilizationOrderIgnored
  ← Achieve [SoundAlarmRaisedWhenMDTMobOrderReceived]
  ← Achieve [FailedMobilizationRecovered]
```

The large number of obstacles and countermeasure goals called for our countermeasure integration and encapsulation techniques. As a result, the countermeasure goals appear to focus on a small number of important goals; e.g., the goal Achieve [IncidentResolvedByAmbulanceIntervention] has 15 exceptions. The overall integration produced 34 exceptions distributed over 7 goals only.

The techniques presented in this paper helped significantly for the following reasons.

Model simplification by separation of concerns. The goals referring to normal situations were systematically distinguished from those handling obstacle occurrences. Emerging assumptions were incrementally down-propagated to obstructed descendants of corresponding anchor goals; this required 7 propagations and produced 28 *Provided* annotations

¹See <http://www.info.ucl.ac.be/~acaillia/publications/las-system.html> for full report.

distributed over 6 goals. Without these annotations the formal specification of those 6 goals would have been cluttered with details related to exceptional cases. Table II quantifies our use of *Provided* annotations.

For example, the goal Achieve [AllocatedAmbulance MobilizationWhenMobilizationOrderPrinted] is defined as follows after integration in the model:

```
Goal Achieve [AllocatedAmbulanceMobilizedWhenMobilization OrderPrinted]
  Provided AllocatedAmbulanceNotLeavingBeforeMobilization
  Provided AllocatedAmbulanceNotUnavailableBeforeMob
  Provided PrintedMobilizationOrderNotIgnored
  Provided MobilizationNotTakenByOtherAmbulance.
```

The full equivalent specification of this goal without *Provided* annotations would completely hide the ideal case; it would then appear fairly hard to distinguish the part of the goal antecedent related to the ideal case from those related to exceptional cases.

The *Detach-Except* operator was applied to the *case-driven* refinement of the goal Achieve [AmbulanceMobilized WhenAllocated]. Allocating an ambulance when not at station was estimated fairly rare – 5% of cases according to typical figures in the domain. The parent goal of these two goals was therefore modified accordingly:

```
Goal Achieve [AmbulanceMobilizedWhenAllocated]
  Except AllocatedAmbulanceNotAtStation
    then Achieve [AllocatedAmbulanceMobilizedOnRoad]
```

Such refactoring reduces model complexity by hiding the part of the model handling the mobilization of an ambulance when on road. The resulting ideal goal model therefore contains fewer refinements and fewer goals, making it easier to understand and clearly separate ideal behaviors from exceptional ones.

Compositionality. Without our techniques, the integration of so many exceptions for only 7 goals would have resulted in large, complex refinements with a combinatorial blow-up of special cases. To illustrate this important point, consider the goal Achieve [AmbulanceMobilizedWhenAllocated]. Its original, ideal specification is:

```
∀amb: Ambulance, inc: Incident
  Allocated(amb, inc)
  ⇒ ◊≤3min ∃amb: Ambulance · Mobilized(amb, inc)
```

After obstacle analysis, this goal is guaranteed through 5 countermeasure goals (see Fig 7). The brute-force integration of only the three countermeasure goals depicted at the bottom of Fig. 6 would have resulted in the following formal specification for the final version of the goal Achieve [AmbulanceMobilizedWhenAllocated]:

```
∀c: UrgentCall, inc: Incident
  Allocated(amb, inc)
  ⇒ ◊≤3min ∃ amb: Ambulance · Mobilized(amb, inc)
    v [ □≥3min ~AmbAvailable(amb, inc)
      → ◊≤6min ∃amb': Ambulance
        amb≠amb' ∧ Mobilized(amb', inc) ]
    v [ □≥3min DisplayedMobilizationIgnored(amb, inc)
      → ◊≤6min Mobilized(amb', inc) ]
    v [ □≥3min PrintedMobilizationIgnored(amb, inc)
      → ◊≤6min Mobilized(amb', inc) ]
```

In addition to this complex specification, the goal refinement structure would have been heavily modified:

```
Achieve [AmbulanceMobilizedWhenAllocated]
  ← Achieve [OtherAmbMobWhenAllocatedAmbUnavailable]
  ← Achieve [AllocAmbMobilizedWhenAmbAvailableUntilMob]
```

```

← Achieve [LateMobWhenDisplayedMobOrderIgnored]
← Achieve [AllocAmbMobilizedWhenAmbAvailableUntilMob
  AndDisplayedMobOrderNotIgnored]
  ← Achieve [LateMobWhenPrintedMobOrderIgnored]
  ← Achieve [AllocAmbMobilizedWhenAmbAvailUntilMob
  AndDisplayedMobOrderNotIgnored
  AndPrintedMobOrderNotIgnored]
  ← ...

```

With such a brute-force integration, each countermeasure goal must be refined by taking other countermeasures into account. This would lead to a combinatorial blow-up of cases. Thanks to our technique, the original specification of this goal and its refinement structure are preserved. The *Except* and *Provided* constructs encapsulate the modifications for a more robust system. Table I provides some figures on goal exceptions for other goals.

No premature decision and freedom of choice. The specification and documentation of exceptional behaviors was separated from the normal ones; this allowed us delaying the decision of how and when the handling of exceptional cases should occur.

Other benefits. The *Replaces* annotation was felt useful for documenting the replacing countermeasure goals –e.g., Achieve [MobilizedAmbInterventionOrMobilizationCancelled] replacing Achieve [MobilizedAmbulanceIntervention] to resolve the obstacle MobilizationCancelled. Without this annotation we would have lost the previous version of the goal.

Exception diagrams significantly helped understand the model where all countermeasures are integrated; they document exceptions one single goal at a time (see Fig. 7). A total of 7 exception diagrams was produced for documenting exceptional cases and countermeasure goals.

Tool support. Our evaluation on the LAS case study was supported by a preliminary tool prototype. Given an obstacle resolution tactic and the corresponding anchor goal, the tool automatically generates the corresponding *Except* or *Replaces* annotations with corresponding countermeasure goal. The *Provided* and *RelaxedTo* constructs are supported as well. The tool also generates exception diagrams.

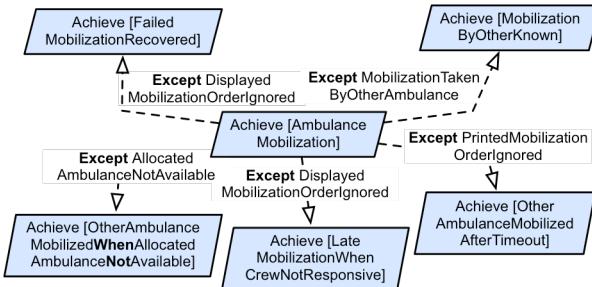


Fig. 7. Exception diagram for Achieve [AmbulanceMobilization]

IX. RELATED WORK

In the *identify-assess-control* cycles of risk analysis at requirements engineering time [8, 13, 14, 17, 20], most of the work so far has been devoted to risk identification and assessment. For risk identification, scenario-based heuristics are available [2, 29] as well as goal-oriented formal techniques [1, 14]. For risk assessment, various kinds of quantitative techniques are available [3, 5, 8, 25]. For risk control, the only work on countermeasure exploration is [14] where the obstacle

TABLE I. USING GOAL EXCEPTIONS

| Goals | Exceptions |
|--|------------|
| Achieve [Incident Resolved By Ambulance Intervention] | 15 |
| Achieve [Ambulance Mobilization] | 6 |
| Achieve [Allocated Ambulance Mobilization When Mobilization Order Printed] | 5 |
| Achieve [Mobilized Ambulance Intervention] | 3 |
| Achieve [Mobilized Ambulance Intervention Or Mobilization Cancelled] | 3 |
| Achieve [Allocated Ambulance Mobilization When Mobilization Order Displayed] | 2 |
| Achieve [Allocated Ambulance Mobilization At Station] | 1 |

TABLE II. USING PROVIDED-CLAUSES

| Goals | Provided |
|--|----------|
| Achieve [Allocated Ambulance Mobilization On Road] | 3 |
| Achieve [Allocated Ambulance Mobilization At Station Based On Location Info] | 4 |
| Achieve [Allocated Ambulance Mobilization When Mobilization Order Printed] | 4 |
| Achieve [Allocated Ambulance Mobilization On Road Based On Location Info] | 3 |
| Achieve [Allocated Ambulance Mobilization When Mobilization Order Displayed] | 3 |
| Achieve [Allocated Ambulance Mobilization At Station] | 1 |

resolution tactics mentioned in this paper are described. We are not aware of any work on systematic integration of countermeasures in a requirement model with a clear, precise semantics.

The relevance and importance of default-based reasoning has been recognized in the context of elaborating requirements or specifications. In [30], a formal framework is proposed for reasoning about evolving requirements. The framework is based on belief revision and default theory; operators for adding and retracting requirements are defined together with formal conditions for their valid application (similarly to our integration operators). The tracing of exceptional requirements is not discussed there. In [24], a specification is structured through axioms and *Overrides* relations. Such relations are derived from the structural decomposition of the system. Specific axioms predominate more general ones when a conflict occurs. This framework comes with formal foundations and well-defined procedures for identifying conflicts and predominance among axioms. It appears more oriented towards specification elaboration. In [27], default specifications are introduced together with exceptions in order to increase the completeness of algebraic specifications; the *But* relation there somewhat corresponds to our *Except* relation.

Our approach mainly differs from those previous efforts in the following directions.

- Our techniques operate at requirements level and benefit from the refinement structure of a goal model. This structure helps in building a model where exception handling is integrated and in propagating required changes throughout the model.
- New requirements for a more robust system are incrementally integrated through obstacle analysis. The model updates are traceable back to the identified obstructed goals and their obstacles.

At programming level, aspects may be used for separating exception handling from normal code [19]. At modelling level, [28] convincingly shows how aspects can be used for separating exceptional behaviors from normal ones. As an alternative to the approach advocated in this paper, robustness

aspects might be incorporated in a goal model by use of constructs similar to the ones sketched in, e.g., [10, 23]. Further work would however be required to define a declarative, logic-based semantics as well as an operational, trace-based semantics for such constructs –which seems unavailable to date. Suitable weaving mechanisms would then need to be defined in this semantic framework.

X. CONCLUSION

The paper presented systematic techniques for integrating countermeasures into ideal goal models. An integration operator was introduced as a model transformation ensuring *progress* towards a more complete model, *minimal change* of the original model, and *refinement correctness preservation*. *Anchor goals* were introduced to define where countermeasure goals should be integrated together with appropriate refinement schemas. Our goal-oriented RE framework was extended with constructs for structuring and documenting exceptional cases. Coming with these, model refactoring operators were proposed enabling analysts to attach and detach exceptions. The approach was evaluated on two case studies, a simple mine pump system and a much more complex ambulance despatching system.

As shown in these case studies, a more complete goal model is obtained while the ideal model is kept visible. The ideal specifications are preserved. The final refinement structure turns out to be nearly the same as the original one. Exceptions are documented aside; analysts and users of the model can dive into independent exceptions one by one. A large number of countermeasure goals can be integrated; the integration techniques reduce model complexity by keeping the combinatorial blow-up of exceptional cases implicit.

The current version of our tool is fairly basic. Among the planned extensions, the increased automation of change propagation deserves highest priority. The propagation procedure itself should be made less dependent on common refinement patterns.

Complementary techniques are needed for selecting “best” countermeasures according to soft goals from the goal model. The responsibilities of agents in exception handling should be integrated as well. Moreover, the use of our exception-related constructs for deriving exception handlers in the corresponding software architecture would be worth investigating. In parallel, their exploitation for runtime self-adaptation in changing contexts appears a promising direction for future work.

ACKNOWLEDGEMENT

This work was supported by the EU Fund for Regional Development & the Walloon Region (TIC-FEDER Grant CE-IQS). We wish to thank B. Lambeau, C. Damas and S. Busard for discussions on our approach, and the reviewers for useful comments.

REFERENCES

- [1] D. Alrajeh, J. Kramer, A. van Lamsweerde, A. Russo and S. Uchitel, “Generating Obstacle Conditions for Requirements Completeness”, *Proc. ICSE’2012: 34th Intl. Conf. Softw. Eng.*, Zürich, May 2012.
- [2] A. Anton and C. Potts, “The Use of Goals to Surface Requirements for Evolving Systems”, *Proc. ICSE’98: Intl. Conf. Softw. Eng.*, May 1998.
- [3] Y. Asnar, P. Giorgini and John Mylopoulos, “Goal-driven Risk Assessment in Requirements Engineering”, *Req. Eng. Jl. 16(2)*, June 2011, 101-116.
- [4] T. Bedford and R. Cooke, *Probabilistic Risk Assessment-Foundations and Methods*. Cambridge University Press, 2001.
- [5] A. Cailliau and A. van Lamsweerde, “Assessing requirements-related risks through probabilistic goals and obstacles”, *Requirements Engineering Journal 18(2)*, Springer-Verlag, 2013, 129-146.
- [6] R. Darimont and M. Lemoine, “Security Requirements for Civil Aviation with UML and Goal Orientation”, *Proc. REFSQ’07: Intl. Conf. on Foundations for Softw. Quality*, LNCS 4542, Springer-Verlag, 2007.
- [7] R. Darimont and A. van Lamsweerde, “Formal Refinement Patterns for Goal-Driven Requirements Elaboration”, *Proc FSE’96: 4th ACM Symp. on the Foundations of Softw. Eng. (FSE’4)*, Oct.1996, 179-190.
- [8] M.S. Feather and S.L. Cornford, “Quantitative Risk-Based Requirements Reasoning”, *Req. Eng. Journal 8(4)*, Springer-Verlag, 2003, 248-265.
- [9] M. Feather, S. Fickas, A. van Lamsweerde, and C. Ponsard, “Reconciling System Requirements and Runtime Behaviour”, *Proc. IWSSD’98 - 9th Intl. Workshop on Soft. Spec. and Design*, Isobe, IEEE, April 1998.
- [10] A. Gil, J. Araújo, “AspectKAOS: integrating early-aspects into KAOS”, *Proc. 15th Workshop on Early Aspects*, ACM, 2009, 31-36.
- [11] M. Joseph, *Real-Time Systems: Specification, Verification and Analysis*, Prentice Hall Intl., 1996.
- [12] A. van Lamsweerde, “Elaborating Security Requirements by Construction of Intentional Anti-Models”, *Proc. ICSE’04, 26th Intl. Conf. on Software Engineering*, ACM-IEEE, May 2004, 148-157.
- [13] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.
- [14] A. van Lamsweerde and Emmanuel Letier, “Handling Obstacles in Goal-Oriented Requirements Engineering”, *IEEE Trans. Softw. Eng. 26(10)*, October 2000, 978-1005.
- [15] E. Letier, *Reasoning about Agents in Goal-Oriented Requirements Engineering*, PhD Thesis, Univ. Cath. Louvain, May 2001.
- [16] N.G. Leveson, *Safeware: System Safety and Computers*. Addison-Wesley, 1995.
- [17] N. Leveson, “An Approach to Designing Safe Embedded Software”, *Proc. EMSOFT 2002 – Embedded Software: 2nd Intl. Conference*, Grenoble, LNCS 2491, Springer-Verlag, October , 2002, 15-29.
- [18] N.G. Leveson, *Engineering a Safer World*. MIT Press, 2011.
- [19] M. Lippert, C. V. Lopes, “A study on exception detection and handling using aspect-oriented programming”, *Proc. ICSE’2000: International Conference on Software Engineering*, IEEE, 2000, 418-427.
- [20] M.S. Lund, B. Solhaug and K. Stølen, *Model-Driven Risk Analysis: the CORAS approach*. Springer-Verlag, 2011.
- [21] R. Lutz, A. Patterson-Hine, S. Nelson, C.R. Frost, D. Tal and R. Harris, “Using Obstacle Analysis to Identify Contingency Requirements on an Unpiloted Aerial Vehicle”, *Req. Eng. Journal 12(1)*, 2007, 41-54.
- [22] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.
- [23] G. Mussbacher, D. Amyot, J. Araújo, A. Moreira, M. Weiss, “Visualizing Aspect-Oriented Goal Models with AoGRL”, *2nd Intl. Workshop on Requirements Engineering Visualization*, IEEE, 2007.
- [24] M. Ryan, “Defaults in Specifications”, *Proc. First IEEE International Symposium on Requirements Engineering*, 1993, 142-149.
- [25] M. Sabetzadeh, D. Falessi, L. Briand, S. Di Alesio, D. McGeorge, V. Ahjem and J. Borg, “Combining Goal Models, Expert Elicitation, and Probabilistic Simulation for Qualification of New Technology, Proc. IEEE 13th Intl. Symp. on High-Assurance Syst. Eng., Nov. 2011, 10-12.
- [26] B. Schneier, *Secrets and Lies: Digital Security in a Networked World*. Wiley, 2004.
- [27] P.-Y. Schobbens, “Exceptions for algebraic specifications: on the meaning of but”, *Sci. Computer Programming 20(1-2)*, 1993, 73-111.
- [28] A. Shaukat, L. Briand, H. Hemmati, “Modeling robustness behavior using aspect-oriented modeling to support robustness testing”, *Software & Systems Modeling 11(4)*, 2012, 633-670.
- [29] A. Sutcliffe, N.A. Maiden, S. Minocha, and D. Manuel, “Supporting Scenario-Based Requirements Engineering”, *IEEE Trans. Software Eng. 24(12)*, Dec. 1998, 1072-1088.
- [30] D. Zowghi and R. Offen, “A Logical Framework for Modeling and Reasoning About the Evolution of Requirements”, *Proc. 3rd IEEE Intl. Symp. on Requirements Engineering*, 1997, 247-257.

Protos: Foundations for Engineering Innovative Sociotechnical Systems

Amit K. Chopra*, Fabiano Dalpiaz†, F. Başak Aydemir‡, Paolo Giorgini‡, John Mylopoulos‡, Munindar P. Singh§

*Lancaster University, Lancaster, UK; email: a.chopra1@lancaster.ac.uk

†Utrecht University, Utrecht, The Netherlands; email: f.dalpiaz@uu.nl

‡University of Trento, Trento, Italy; email: {aydemir, pg, jm}@disi.unitn.it

§North Carolina State University, Raleigh, NC, USA; email: m.singh@ieee.org

Abstract—We address the challenge of requirements engineering for *sociotechnical* systems, wherein humans and organizations supported by technical artifacts such as software interact with one another. Traditional requirements models emphasize the goals of the stakeholders above their interactions. However, the participants in a sociotechnical system may not adopt the goals of the stakeholders involved in its specification. We motivate, Protos, a requirements engineering approach that gives prominence to the interactions of autonomous parties and specifies a sociotechnical system in terms of its participants' social relationships, specifically, commitments. The participants can adopt any goal they like, a key basis for innovative behavior, as long as they interact according to the commitments. Protos describes an abstract requirements engineering process as a series of refinements that seek to satisfy stakeholder requirements by incrementally expanding a specification set and an assumption set, and reducing requirements until all requirements are accommodated. We demonstrate this process via the London Ambulance System described in the literature.

Index Terms—Requirements, refinement, architecture, commitments, teams, protocols

I. INTRODUCTION

We define a *SocioTechnical System* (STS) as one involving interactions between humans and organizations (*principals*) facilitated by *technical artifacts*, including software. STSs under this definition include two or more autonomous parties, who ordinarily act and interact in ways that promote their respective agendas. Thus the participants may act in ways that are unexpected by others.

We further restrict our attention to STSs whose rules of encounter are formally represented, whether created via explicit engineering or otherwise. The benefit of an “institutionalized” STS is that its explicit rules of encounter facilitate reasoning by (prospective) participants about whether and how to participate in the STS. Examples of such STSs can be found in many domains, such as healthcare, finance, transportation, and commerce. STSs, especially those with explicit rules of encounter, provide a conceptual basis for innovation by the participants. (Other STSs may also support innovation but we limit our claims to institutional STSs.)

Innovations in STSs include the emergence of new social practices. For example, beginning from goals of passengers and transporters, we might design an airport as a hub where they can execute their transactions. However, an enterprising person could invent the notion of an airport as a venue

for shopping in general (as Brendan O'Regan did in 1947). Similarly, the goals of providing capital for new ventures yield a market for public offerings of stocks, which have morphed into the financial systems of today. Notice that not every innovation is desirable and in general some participants would gain and some would lose from any innovation.

Classical Requirements Engineering (RE) approaches fail to sufficiently support innovation and thus do not help realize the innovative potential of STSs. Classical RE begins from an elicitation of the goals of stakeholders, followed by an analysis phase that produces a specification of a software system that would satisfy those goals given some assumptions about its operating environment. A limitation of such approaches is that the goals of the stakeholders used as a basis for modeling may have little in common with the goals of the participants in the STS (whom we call “principals”). We posit that a goal-based treatment of STSs undercuts innovation. It offers up the following dilemma. Either the participants innovate but in an ad hoc manner (such as on social media today, where memes and conventions such as emoticons emerge sporadically) or the participants are regimented to the original goals and do not innovate at all. Regimentation is not viable for autonomous participants. In practice, ad hoc innovation is what we see most often.

Motivation. Our research question is as follows: *How can RE facilitate innovation in STSs? That is, how may the stakeholders of an STS systematically produce a specification that facilitates innovation?*

We posit that the RE effort be broken into two logically distinct phases: (1) coming up with the rules of encounter in an STS and (2) given the rules of encounter, coming up with models of participants (e.g., their policies) that determine how they participate. A common representation of the rules of encounter ties these two phases together. The first phase is carried out jointly by (or on behalf of) the stakeholders specifying the STS. The second phase is carried out separately by (or on behalf of) stakeholders for each participant.

A major benefit of the aforementioned common representation is that not only does it enable the interoperation needed to realize the STS but also decouples the participants with respect to what is not specified, and thus frees them up to innovate. As long as they comply with the rules of encounter, the innovation is not ad hoc.

STSs are exemplified by many practical settings. Let us consider a healthcare system, which we understand as involving interactions among physicians, nurses, patients, hospitals, insurance sellers, and regulatory bodies. There are two potential ways one can approach the requirements engineering of such a system given stakeholder requirements. The traditional or *regimented* approach is to specify a software module that all the principals would use. The other or *interaction-oriented* approach involves is to specify the interaction protocols that would support meeting the requirements.

An interaction protocol would describe how any principal adopting one or more *roles*, such as hospital or physician, would interact with others. In particular, the protocol would specify the social expectations that principals playing one of the roles could have of principals playing other roles. Each principal would be free to develop its own software in accordance with its requirements. For example, different principals playing the role of hospital could adopt different implementations and policies by which they conduct their business. We refer to each principal's software as its *agent*.

Traditional works on requirements engineering (RE) and methodologies [1], [2] generally tend to be regimented: they address the specification of software conceptualized as a machine that resides within STSs, not the sociotechnical system itself. Approaches such as Tropos [3] begin from modeling the sociotechnical system, but end up with a regimented software system.

We refine the above question further to emphasize the foundational aspects of RE, namely, *What is a suitable formulation and formalization of the STS design space that is conducive to the autonomy both of stakeholders and principals?* To focus on the representational aspects, we place as out of our scope the important challenges of negotiation among stakeholders and of collaborative and concurrent engineering of an STS.

Contributions. We make the following contributions:

- We introduce a way of specifying an STS as a protocol in terms of roles and their social commitments [4] to each other. Our approach applies to social expectations generally, though for concreteness, we focus on commitments here.
- We present a model for requirements satisfaction in which stakeholder requirements are satisfied by a protocol specification, modulo any stated assumptions.
- We propose a generalization of one of the foundational axioms of RE that accommodates the separation of concerns between specifying an STS and an individual principal specifying its agent.
- We present an abstract design process for STSs that extends the classical idea of refinement. We refer to this extended notion of refinement as *social refinement* to emphasize the refinement of requirements into commitment-based specifications. We illustrate the process in a systematic case study of the London Ambulance System.

Organization. The rest of the paper is organized as follows. Section II introduces background on protocols and require-

ments refinement. Section III revisits the traditional RE problem to accommodate STSs and shows that specifying STSs and specifying agents are two independent problems. Section IV motivates and formalizes a set of refinement reductions to obtain STS specifications from stakeholders' requirements. Section V applies our approach to the London Ambulance System. Section VI discusses related work and Section VII summarizes our contributions, and outlines future directions.

II. BACKGROUND

We adopt a scenario from automobile insurance (based on Browne and Kellet's [5] real-life description).

A. Classical Formulation of Requirements

Zave and Jackson [1] characterize RE in terms of A (a set of domain assumptions), M (a software (machine) specification), and R (a set of stakeholder requirements). A requirements engineer's task is to come up with a software specification and the domain assumptions that together guarantee that the requirements are met, which Eq. 1 shows:

$$A, M \vdash R \quad (1)$$

B. Commitment Models

A commitment represents a directed social expectation between principals. We express a commitment as a four-tuple $C(\text{debtor}, \text{creditor}, \text{antecedent}, \text{consequent})$: the debtor is committed to the creditor that if the antecedent holds, the consequent will hold [6]. A commitment is detached when its antecedent holds, unless it has timed out. It is discharged when its consequent holds, unless it has timed out.

Example 1. The insurance company commits to the car owner to reimburse any damages if the insurance policy is valid: $C(\text{INSURER}, \text{CAR OWNER}, \text{hasValidInsurance}, \text{getReimbursed})$.

Example 2. If the CAR OWNER has a valid policy, the above commitment is detached and the following unconditional commitment holds: $C(\text{INSURER}, \text{CAR OWNER}, \text{true}, \text{getReimbursed})$.

If the car owner is reimbursed, the insurer's commitment is discharged. The commitment is violated if the car owner owns a valid policy but does not get reimbursed for damages.

A *protocol* specifies the rules of encounter among principals adopting roles in it. In general, a commitment is established by communication from its debtor to its creditor: it is thus autonomously created and becomes part of the social state of the interacting parties. However, we concentrate here on the commitments themselves without regard to the communications that bring them about. Thus, here, a protocol specifies a set of commitments, such as in Example 1. By adopting roles in a protocol, a principal would become the creditor of some commitments and debtor of others.

Commitments, like goals, are a high-level abstraction. However, unlike goals, commitments provide a standard of correctness for interactions among principals that is independent of their mental states. Thus, for example, a particular insurance company may intend to avoid paying for the damages of a

particular car owner even if the owner has valid coverage. If the insurance company goes through with its intention, though, it would violate its commitments to the car owner.

C. Refinement

Refinement is a foundational concept in software engineering [7]. Goal refinement, in particular, is a fundamental concept for systematically extracting a specification from a set of requirements, e.g., in Tropos.

Design refinement is a relation between design problems p and p' , where p' is an incremental improvement of p , $p \hookrightarrow p'$, and a solution for problem p' also constitutes a solution for problem p . A *design space* $(P, R_{\hookrightarrow}, p_0)$ consists of an abstract set of design problems, P ; a root problem, $p_0 \in P$; and a refinement relationship $R_{\hookrightarrow} \subseteq P \times P$. A root problem is one that is not a refinement of any other problem.

In Zave and Jackson's terminology an engineer could begin from the requirements in R and progressively refine them to produce an implementable M . Software development, then, is concerned with implementing M .

III. RE FOR SOCIOTECHNICAL SYSTEMS

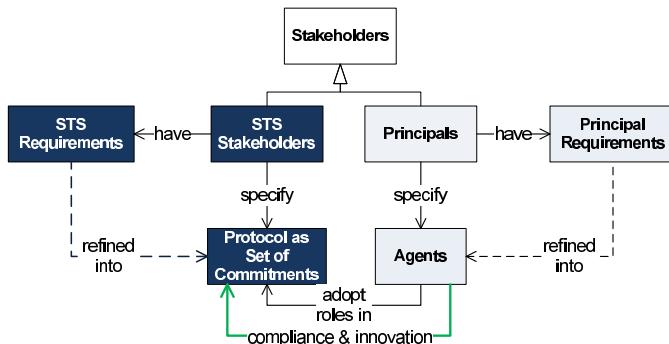


Fig. 1: The overall design space for sociotechnical systems.

Figure 1 illustrates the overall design space for STSs as discussed in Section I. Stakeholders are of two kinds: STS stakeholders and principals (that is, runtime participants in the STS). The principals of an STS might have been involved as STS stakeholders in its design, but that is not necessary. The innovation opportunity of STSs arises partly because its principals may never have been imagined as its stakeholders during design, though the principals should have something in common with the STS stakeholders. The dark boxes (and relationships) capture the design space for STS stakeholders. The light boxes (and relationships) capture an individual principal's design space. Whereas STS stakeholders specify a protocol by refining STS requirements, a principal specifies an agent from its requirements. The principal, via its agent, adopts one or more roles in the protocol. By decoupling STS requirements from principal requirements and concomitantly decoupling protocol (STS) specifications from agent specifications, we enable a principal to comply or not with a protocol. Additionally, a principal may comply with a protocol while functioning in novel ways, thereby promoting innovation.

Below, for clarity, we identify STS stakeholders with roles (based on the intuition that stakeholder types and STS roles would normally coincide) and use proper names to identify principals. Thus, for example, in an automobile insurance system, the stakeholders include INSURER, MECHANIC, and CAR OWNER. Alessia and Cristina are particular mechanics; Great Insurance Co. is a particular insurance company.

A. Rethinking RE Characterization

The conception of the design space in Figure 1 necessitates a rethinking of Zave and Jackson's formulation. Eq. 1 is adequate for refining the stakeholders' requirements into a specification of a machine, i.e., the software that would reside in an STS. For example, we might specify software that would support a car owner to file a claim and an insurance staff member to approve the claim and assign a mechanic. The formulation captures the essence of regimented approaches, as illustrated by Example 3.

Example 3. Let stakeholders INSURER and MECHANIC get together to design software for an automobile insurance STS (we neglect CAR OWNER for simplicity). Let R_1 be the set of INSURER's requirements, containing only one requirement: any replacement parts ordered by a mechanic that are priced above \$500 must be approved by the INSURER. This requirement, following Eq. 1, would be refined into a software specification that disables for mechanics the functionality of ordering parts priced over \$500 unless INSURER's approval is recorded. The specification can be implemented using an access control module.

If a principal who plays MECHANIC wants to order parts without the insurance company's approval (e.g., because of urgency or because the approver—(staff member)—is on holiday), the above-mentioned software would not allow the mechanic to proceed. The mechanic would be forced to wait for approval—the software, effectively, regiments interactions [8].

What we would like instead is to come up with a specification that captures the essence of the stakeholder requirement and yet does not regiment the actions of any principal in the STS. This is possible if stakeholder requirements are refined into, not a machine, but a protocol specification. Any principal who adopts a role in the protocol is free to specify (and implement) its own agent in accordance with its own requirements. Eq. 2 captures the above intuition. Ag and S are the sets of agent and protocol specifications, respectively. Eq. 2 separates the essential nature of the interaction in the protocol from the autonomous decision-making of the participants involved in its enactment.

$$A, Ag, S \vdash R \quad (2)$$

Note that when Ag is a singleton and S is thus a null protocol, Eq. 2 reduces to Eq. 1. Therefore, Eq. 2 generalizes the traditional requirements specification problem equation, Eq. 1, to a setting of multiple autonomous principals. Example 4 illustrates Eq. 2.

Example 4. Consider R_1 from Example 3. Let S_1 be the protocol specification $\{C(MECHANIC, INSURER, \text{price}(item) > 500, \text{getApproval}(item) \text{ precedes } \text{order}(item))\}$. For simplicity, let A_1 (the set of assumptions) be empty. Let $Ag_1 = \{i, m\}$; and i and m be the agents of Great Insurance Co. and Alessia, respectively. Suppose Alessia has designed m to obtain approvals before ordering any parts. The overall system is correct, that is, $A_1, Ag_1, S_1 \vdash R_1$.

Example 5. Modify Example 4 such that Alessia's agent m never asks for preapproval, so it would violate the above-mentioned commitment to obtain preapproval. Let's refer to this agent as m' and let $Ag'_1 = \{i, m'\}$. Then R_1 wouldn't be satisfied either, i.e., $A_1, Ag'_1, S_1 \not\vdash R_1$.

B. Modularity of the Design Space

We can achieve the separation of design spaces illustrated in Figure 1. A protocol provides a solution for the requirements, independently of the agents who would ultimately play roles in that protocol to instantiate the STS. That is, given some assumptions A_S of the domain, protocol S ensures R under the assumption E that all principals adopting roles in the protocol satisfy their commitments. In other words, the following holds.

$$A_S, E, S \vdash R \quad (3)$$

Example 6. Returning to our example, under the assumption E (here meaning that any principal playing MECHANIC would satisfy the commitment to get approval), $A_1, E, S_1 \vdash R_1$.

Assuming E during STS design does not mean that any principal who wants to play a role in the STS must be compliant. Its purpose is to capture precisely the intuition that it is the *satisfaction* of the commitment (and not, for instance, its *creation*) that satisfies some requirement. Also note that E is not relativized to the particular STS (that is why there is no E_1 as there are R_1 , S_1 , and A_1). The assumption E is purely formal: it assumes the satisfaction of whichever commitments there are in the specification; in particular, assuming E is not a matter of choice for the stakeholders.

We are concerned with the process of designing protocols from requirements, as in Eq. 3 (and the dark boxes in Figure 1). Example 7 illustrates how agent specifications, as in the lighter boxes in Figure 1, fit into the overall requirements satisfaction argument.

Example 7. We know that $A_1, E, S_1 \vdash R_1$. Cristina wants to play MECHANIC in S_1 . Cristina's requirements are R_c and she wants to design an agent c accordingly. R_c may or may not contain requirements imposed by the protocol. Let's say R_c contains the requirements and c is specified appropriately following Eq. 1, that is, $A_c, c \vdash R_c$. As before, let i (Great Insurance Co.'s agent) play INSURER and, further, that $A_i, i \vdash R_i$. Putting everything together, we get $A_i, A_c, A_1, \{i, c\}, S_1 \vdash R_1$. This is like Eq. 2: $A = A_i \cup A_c \cup A_1$; $Ag = \{i, c\}$, and $S = S_1$. (The assumption E does not figure in the final equation as it is “replaced” by actual agent specifications.)

Example 7 demonstrates that even the simple modification of introducing a protocol yields interesting payoffs. The protocol provides a precise description of the nature of the interactions among principals but leaves open the possibility of enacting the protocol in multiple ways based on their own requirements, thus enabling innovation. For example, this approach accommodates heterogeneous software for different mechanics, and enables ordering before getting the approval (although a mechanic who does so may lose money if the insurance company denies the authorization).

IV. SOCIAL REFINEMENT

We seek a theory of design for STSs founded on the concept of refinement. Since the concepts in terms of which STSs are conceived (e.g., roles, protocols, and commitments) are fundamentally different from those of traditional software engineering (e.g., goals, functions, and actions for requirements, statements and variables for programs), our task is to define new refinements that are specific to STS design problems.

Below, we introduce our model elements, use them to define a design configuration, and introduce a design process.

A. Model Elements

The following are the key primitives of Protos.

Proposition, which represents a state of the world in the domain of the STS. A proposition may be atomic or composite (representing the conjunction or disjunction of propositions). The set of all propositions is \mathcal{P} .

Stakeholder, an autonomous entity present during the design process of an STS.

Team, one or more stakeholders who function as a cohesive unit for design purposes. That is, no team is empty. We denote that τ_i is a subteam of τ as $\tau_i \sqsubseteq \tau$. We write a team τ that comprises subteams τ_i as $\tau = \bigsqcup_i \tau_i$. We do not consider the subtleties of organizational structures, though our approach could be enhanced to incorporate such models. The set of all teams is \mathcal{T} , and includes individual stakeholders as unary terms.

Commitment, a social relationship between a debtor team and a creditor team, referring to an antecedent proposition and a consequent proposition. The set of all commitments is \mathcal{C} . Thus, $\mathcal{C} \subseteq \mathcal{T} \times \mathcal{T} \times \mathcal{P} \times \mathcal{P}$.

Refinement, a reflexive, transitive, antisymmetric relation between propositions. We notate refinement as \hookrightarrow where $a \hookrightarrow b$ means that a refines into b (that is, b is a refinement of a). Thus, $\hookrightarrow \subseteq \mathcal{P} \times \mathcal{P}$. The refinement of commitments and other constructs follows from the above. We lift refinement to design configurations where one configuration refines into another.

Conflict, an irreflexive, symmetric relationship over propositions. We notate conflict as \oplus where $p \oplus q$ means that p conflicts with q . Thus, $\oplus \subseteq \mathcal{P} \times \mathcal{P}$.

Requirement, a representation of an expectation that a team would like to achieve a proposition. The set of all requirements is \mathcal{R} . $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{P}$. The notation $R(\tau, \pi)$ means that $(\tau, \pi) \in \mathcal{R}$.

Onus, a representation of the assumption that a team takes on the onus of ensuring a proposition. $O(\tau, \pi)$ means that team τ takes on the onus for ensuring proposition π . We include onus assertions with other assumptions.

B. Design Configurations

We use these model elements to talk about design episodes. A design episode proceeds from one design configuration to another by systematically applying refinements. We define the following concepts to help us describe design episodes.

Requirements are given as R , a finite set of assertions describing what each team (stakeholder) wants to achieve in the STS. The teams are autonomous, each holding a stake in its own requirements, though a team's requirements may hold for its subteams. Thus, $R \subseteq \mathcal{R}$.

Specifications are given as S , a finite set of assertions describing how the STS to be will function. As motivated above, these assertions describe the interactions in the STS but not the internal details of any of its participants. The interactions are naturally captured as social relationships among the participants. Thus, $S \subseteq \mathcal{C}$.

Domain assumptions are given as A , a finite set of assertions that must hold true in order to ensure that the specifications will satisfy the requirements. The set of possible domain assumptions is $\mathcal{A} = \{p | p \in \mathcal{P}\} \cup \{O(\tau, p) | \tau \in \mathcal{T} \text{ and } p \in \mathcal{P}\} \cup \{a \rightarrow b | a, b \in \mathcal{P}\}$.

Needs are given as N , a finite set of requirements. That is, $N \subseteq \mathcal{R}$. N represents the set of requirements that are yet to be addressed during the design episode.

The foregoing leads to a definition of a design configuration.

Definition 1. Given a set of stakeholder teams \mathcal{T} and a set of propositions \mathcal{P} , a design configuration is a tuple $\langle S, A, N \rangle$, where $N \subseteq \mathcal{R}$; $S \subseteq \mathcal{S}$; and $A \subseteq \mathcal{A}$.

Table I contains an illustration of the complete process of deriving specifications from requirements for an automobile insurance scenario. C, I, M, G, and F represent the stakeholders—CAR OWNER, INSURER, MECHANIC, MANAGER, and FINANCE, respectively. MANAGER and FINANCE are subteams of INSURER; the former approves the payments and the latter makes the transactions. In Table I, the S, A, and N cells of each row constitute a design configuration. We refer to the table to illustrate our formal definitions.

The following definitions are needed in Section IV-C to specify how a design process may begin, terminate, and whether its outcome is consistent and meets the requirements.

Definition 2. A design configuration $\langle S, A, N \rangle$ is initial for requirements R if and only if $N = R$, $S = \emptyset$, and $A = \emptyset$.

In Table I, Row 1 refers to the initial design configuration. CAR OWNER's requirement is to be prepared for the emergencies: $R(c, \text{prepared})$, INSURER has the requirement of selling insurance: $R(I, \text{sold})$, and MECHANIC wants to get paid for his repair services.

Definition 3. A design configuration $\langle S, A, N \rangle$ is final if and only if $N = \emptyset$.

In Table I, Row 8 refers to the final design configuration.

Definition 4. A design configuration $\langle S, A, N \rangle$ is consistent if and only if $A \cup S \not\vdash \text{false}$.

Definition 5. A design configuration $\langle S, A, N \rangle$ satisfies requirements R if and only if $A \cup S \vdash R$.

C. Design Process

The requirements of stakeholders feed a design (refinement) process, which we imagine as being conducted by stakeholders and facilitated by requirements engineers. The output of the process is an STS specification, and a set of domain assumptions. We model the design process as iteratively taking a design configuration and producing another, refined, design configuration through an application of one of the above reductions, beginning from an initial configuration and ending in a final configuration.

We initialize a design configuration from the requirements by treating each requirement as the technical debt of the relevant team. The design process iteratively addresses each need. A team may take on the onus for any of its needs. Alternatively, it may obtain a commitment from another team, in which case its need would change to the antecedent of the commitment. A design episode concludes when a configuration is obtained that resolves all needs of all teams.

Potentially, more than one reduction may apply on a given design configuration. That is, the design space can be large. Some explorations of it may end up in failure. An exploration is consistent when it constitutes a series of refinements from an initial to a final consistent configuration.

Definition 6. A design step takes as input a configuration $\langle S, A, N \rangle$ and produces a configuration $\langle S', A', N' \rangle$ provided we can conclude $\langle S, A, N \rangle \rightarrow \langle S', A', N' \rangle$.

In Table I, going from one row to the next is a design step (the latter row is annotated with the applied reduction from Section IV-D).

Definition 7. A design path for requirements R is a finite series of configurations $\langle S_0, A_0, N_0 \rangle \dots \langle S_n, A_n, N_n \rangle$ where (1) $\langle S_0, A_0, N_0 \rangle$ is initial for R ; (2) $\langle S_n, A_n, N_n \rangle$ is final and consistent; and (3) for each i , $0 \leq i < n$, $\langle S_i, A_i, N_i \rangle \rightarrow \langle S_{i+1}, A_{i+1}, N_{i+1} \rangle$ is a design step.

Table I shows a design path from Row 1 to Row 8.

Definition 8. A design path $\langle S_0, A_0, N_0 \rangle \dots \langle S_n, A_n, N_n \rangle$ for requirements R is sound if and only if $\langle S_n, A_n, N_n \rangle$ is consistent and $\langle S_n, A_n, N_n \rangle$ satisfies R .

In Table I, the design path from Row 1 to Row 8 is sound. Soundness relies on Theorem 1 that establishes that design paths following the reductions below are sound.

Recall Eq. 3 states that $A_S, E, S \vdash R$. It captures the problem of specifying a protocol from requirements. The

TABLE I: Illustrating refinement on the insurance scenario

| | Specification (S) | Assumptions (A) | Needs (N) | Refinement type |
|---|--|--|---|--------------------------------|
| 1 | \emptyset | \emptyset | $R(C, prepared), R(I, sold), R(M, paid)$ | |
| 2 | \emptyset | $A_1 = \{prepared \leftrightarrow covered \wedge eme\}$ | $R(C, covered \wedge eme), R(I, sold), R(M, paid)$ | Need refinement |
| 3 | $C(C, I, covered, sold), C(I, C, sold, covered)$ | A_1 | $R(C, eme \wedge sold), R(I, covered), R(M, paid)$ | Cyclic commitments |
| 4 | $C(C, I, repaired, data \wedge fee), C(I, C, data \wedge fee, repaired)$ | $A_2 = A_1 \cup \{sold \leftrightarrow data \wedge fee, covered \leftrightarrow repaired\}$ | $R(C, eme \wedge data \wedge fee), R(I, repaired), R(M, paid)$ | Commitment refinement <i>i</i> |
| 5 | $C(C, I, found \wedge fixed, data \wedge fee), C(I, C, data \wedge fee, found \wedge fixed)$ | $A_3 = A_2 \cup \{repaired \leftrightarrow found \wedge fixed\}$ | $R(C, eme \wedge data \wedge fee), R(I, found \wedge fixed), R(M, paid)$ | Commitment refinement <i>i</i> |
| 6 | $C(C, I, found \wedge fixed, data \wedge fee), C(I, C, data \wedge fee, found \wedge fixed), C(M, I, paid, fixed), C(I, M, fixed, paid)$ | A_3 | $R(C, eme \wedge data \wedge fee), R(I, found \wedge paid), R(M, fixed)$ | Cyclic commitments |
| 7 | $C(C, I, found \wedge fixed, data \wedge fee), C(I, C, data \wedge fee, found \wedge fixed), C(M, I, paid, fixed), C(I, M, fixed, paid)$ | $A_4 = A_3 \cup \{G \sqsubseteq I, F \sqsubseteq I, paid \leftrightarrow transfer \wedge app\}$ | $R(C, eme \wedge data \wedge fee), R(I, found), R(M, fixed), R(G, app), R(F, transfer)$ | Subteams |
| 8 | $C(C, I, found \wedge fixed, data \wedge fee), C(I, C, data \wedge fee, found \wedge fixed), C(M, I, paid, fixed), C(I, M, fixed, paid)$ | $A_5 = A_4 \cup \{O(C, eme \wedge fee \wedge data), O(I, found), O(M, fixed), O(G, app), O(F, transfer)\}$ | \emptyset | Onus |

design process described above conforms to Eq. 3. S_n and A_n in the design path for R map to S and $A_S \cup E$, respectively. If the design path is sound, we obtain the relation $S_n, A_n \vdash R$.

D. Social Refinement Types

Below, we list social refinement types supported by Protos. (Here the set operators associate to the left and to reduce clutter we do not place assertions in quotations.)

Need refinement: Based on proposition refinement. The intuition is that if p refines to p' , a need for p can be met by meeting a need for p' . For soundness, we must record the assumption that p refines to p' .

$$\begin{aligned} \langle S, A, N \cup \{R(\tau, p)\} \rangle \hookrightarrow \\ \langle S, A \cup \{p \leftrightarrow p'\}, N \setminus \{R(\tau, p)\} \cup \{R(\tau, p')\} \rangle \end{aligned}$$

Notice that we write the needs set as $N \cup \{R(\tau, p)\}$ to ensure that a need $R(\tau, p)$ belongs to the needs set. In the resulting configuration, we remove $R(\tau, p)$ from N to make sure it is not present in the resulting needs set and introduce the refined need $R(\tau, p')$ explicitly.

Example 8. Row 2 in Table I is obtained from Row 1 by refining $R(C, prepared)$ into two other needs, $R(C, covered)$ and $R(C, eme)$, namely, accident coverage and emergency response, respectively.

Commitment introduction i: If τ_1 has a need for q , τ_1 can address that need by obtaining a commitment from τ_0 to τ_1 whereby τ_0 commits to bringing about q provided p holds. Here, τ_1 takes on the need for p .

$$\begin{aligned} \langle S, A, N \cup \{R(\tau_1, q)\} \rangle \hookrightarrow \\ \langle S \cup \{C(\tau_0, \tau_1, p, q)\}, A, N \setminus \{R(\tau_1, q)\} \cup \{R(\tau_1, p)\} \rangle \end{aligned}$$

Commitment introduction ii: If τ_1 has a need for q , τ_1 can address that need by obtaining a commitment from τ_0 to τ_1 whereby τ_0 commits to bringing about q provided p holds. In

this case, we add an assumption that p will hold.

$$\begin{aligned} \langle S, A, N \cup \{R(\tau_1, q)\} \rangle \hookrightarrow \langle S \cup \{C(\tau_0, \tau_1, p, q)\}, \\ \{A \cup \{p\}\}, N \setminus \{R(\tau_1, q)\} \rangle \end{aligned}$$

Commitment refinement i: Based on the refinement of either or both of its antecedent and consequent. That is, if the antecedent or consequent of a commitment can be refined, then so can the commitment. As before, we record the refinements as assumptions.

$$\begin{aligned} \langle S \cup \{C(\tau_0, \tau_1, p, q)\}, A, N \cup \{R(\tau_1, p)\} \rangle \hookrightarrow \\ \langle S \setminus \{C(\tau_0, \tau_1, p, q)\} \cup \{C(\tau_0, \tau_1, p', q')\}, \\ A \cup \{p \leftrightarrow p', q \leftrightarrow q'\}, N \setminus \{R(\tau_1, p)\} \cup \{R(\tau_1, p')\} \rangle \end{aligned}$$

Notice that, the resulting commitment need not logically entail the original commitment. For example, a commitment to provide coffee for payment may be refined into a commitment to provide coffee for payment of Euros, though the second commitment is weaker than the original.

Example 9. Row 4 in Table I is obtained by refining the commitments in Row 3. Specifically, buying a policy is refined into providing personal data and paying the fee, whereas providing coverage is refined into repairing the car. The commitments and needs in Row 4 reflect this refinement.

Commitment refinement ii: Based on the refinement of either or both of the commitment's creditor and debtor.

$$\begin{aligned} \langle S \cup \{C(\tau_0, \tau_1, p, q)\}, A, N \cup \{R(\tau_1, p)\} \rangle \hookrightarrow \\ \langle S \setminus \{C(\tau_0, \tau_1, p, q)\} \cup \{C(\tau_0', \tau_1', p, q)\}, \\ A \cup \{\tau_0' \sqsubseteq \tau_0, \tau_1' \sqsubseteq \tau_1\}, \\ N \setminus \{R(\tau_1, p)\} \cup \{R(\tau_1', p)\} \rangle \end{aligned}$$

Cyclic commitments: Given n teams ($n \geq 2$) where for each i , $0 \leq i < n$, team τ_i has a need for p_i , these teams can address their needs via (cyclic) commitments from τ_i to $\tau_{(i+1 \bmod n)}$ whereby τ_i commits to bringing about $p_{(i+1 \bmod n)}$ provided p_i holds. Reciprocal commitments are a special case of cyclic

commitments when $n = 2$.

$$\begin{aligned} \langle S, A, N \cup \bigcup_i \{R(\tau_i, p_i)\} \rangle \hookrightarrow \\ \langle S \cup \bigcup_i \{C(\tau_i, \tau_{(i+1 \bmod n)}, p_i, p_{(i+1 \bmod n)})\}, A, \\ N \setminus \bigcup_i \{R(\tau_i, p_i)\} \cup \bigcup_i \{R(\tau_{(i+1 \bmod n)}, p_i)\} \rangle \end{aligned}$$

Example 10. Row 3 in Table I is obtained from Row 2 by applying Cyclic Commitments. In Row 2, CAR OWNER and INSURER need coverage and sale of policies, respectively. To meet these needs, in Row 3, CAR OWNER commits to buying a policy if INSURER provides coverage for accidents and INSURER commits to providing coverage if a policy is bought. Further, CAR OWNER and INSURER need to buy policy and provide coverage, respectively.

Subteams: If a team τ has a need p , we can assign p_i to subteams τ_i ; this is appropriate only because we assume that τ_i is a subteam of τ .

$$\begin{aligned} \langle S, A, N \cup \{R(\tau, p)\} \rangle \hookrightarrow \\ \langle S, A \cup \{p \hookrightarrow \bigwedge_i p_i\} \cup \bigcup_i \{\tau_i \sqsubseteq \tau\}, \\ N \setminus \{R(\tau, p)\} \cup \bigcup_i \{R(\tau_i, p_i)\} \rangle \end{aligned}$$

Example 11. Row 7 in Table I is obtained by applying Subteams. The payment need $R(I, paid)$ is refined into approval and transaction, namely, $R(I, app)$ and $R(I, transfer)$. The manager and the finance department, which are the insurer's subteams, adopt these needs, that is, $R(G, app)$ and $R(F, paid)$.

Onus: A team takes on the onus for some need locally; that is, it decides not to delegate that need to another team.

$$\begin{aligned} \langle S, A, N \cup \{R(\tau, p)\} \rangle \hookrightarrow \\ \langle S, A \cup \{O(\tau, p)\}, N \setminus \{R(\tau, p)\} \rangle \end{aligned}$$

Example 12. Row 8 is obtained by applying Onus: all stakeholders take on the onus for their remaining needs.

Composition: Refinements compose in that, if part of a configuration is refined through a particular operation, so is an entire configuration through the same operation.

$$\begin{aligned} \langle S', A', N' \rangle \hookrightarrow \langle S'', A'', N'' \rangle & \quad (S \cup S' \cup A \cup A') \not\vdash \text{false} \\ \langle S \cup S', A \cup A', N \cup N' \rangle \hookrightarrow \langle S \cup S'', A \cup A'', N \setminus N' \cup N'' \rangle \end{aligned}$$

E. Logic of Design Elements

For simplicity, we assume that a logic is available for reasoning about the various elements of a design configuration. In particular, this logic provides an inference relation, notated \vdash . The various elements of a design configuration are closed under \vdash , as specified by Table II.

F. Example of Design Paths

Figure 2 illustrates the above definitions. The root is the requirement. Immediately below the root is the initial design configuration. Each of the leaves is a final configuration and satisfies the requirements. Each configuration is consistent.

TABLE II: The underlying logic is propositional logic augmented with the following axioms pertaining to the symbols introduced in Protos.

| | |
|---|---|
| Conflict means we cannot satisfy both | $\frac{p \quad p \oplus q}{\neg q}$ |
| A subteam's stronger need satisfies a team's need: AND | $\frac{R(\tau', p \wedge q) \quad \tau' \sqsubseteq \tau}{R(\tau, p)}$ |
| A subteam's stronger need satisfies a team's need: OR | $\frac{R(\tau', p) \quad \tau' \sqsubseteq \tau}{R(\tau, p \vee q)}$ |
| Subteams together cover needs that a team's need refines to | $\frac{\bigwedge_i R(\tau^i, p_i) \quad p \hookrightarrow \bigwedge_i p_i \quad \bigwedge_i \tau^i \sqsubseteq \tau}{R(\tau, p)}$ |
| If a team takes on an onus, the corresponding need is covered | $\frac{O(\tau, p)}{R(\tau, p)}$ |
| A conditional commitment along with its antecedent cover its consequent | $\frac{R(\tau, p) \quad C(\tau', \tau, p, q)}{R(\tau, q)}$ |
| An unconditional commitment covers its consequent | $\frac{C(\tau', \tau, \text{true}, q)}{R(\tau, q)}$ |

Each path is a well-formed design path and is sound for the requirements. The edge labels are informal descriptions.

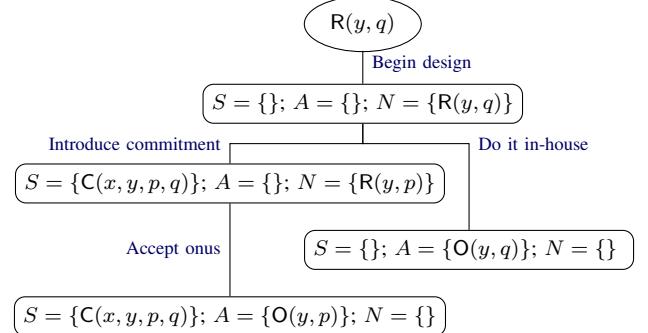


Fig. 2: Paths through the design space.

G. Soundness

We now establish the soundness of our formal model.

Theorem 1 (Soundness). *Let $P = \langle S_0, A_0, N_0 \rangle \dots \langle S_n, A_n, N_n \rangle$ be a design path for requirements R . Then P is sound.*

PROOF SKETCH. Establish the invariant that $A \cup S \cup N \vdash R$ by structural induction: it holds for an initial configuration by construction and for each subsequent configuration by inspection of the reductions. In a final configuration, $N = \emptyset$: hence we have the result.

The above reductions do not consider conflict. The following reduction ensures that if τ_1 and τ_2 (could be the same

TABLE III: A portion of the design process during the modeling session on the London Ambulance System

| Step | Specification | Assumptions | Needs | Refinement |
|------|---|--|--|-----------------------------------|
| 1 | $C(CT, SC, address \wedge status, incidentTaken)$ | $A_1 = \{incidentReported \hookrightarrow address \wedge status, O(SC, address \wedge status), O(CT, incidentTaken)\}$ | $R(SC, ambReceived), R(AC, mobilityInfoSent), R(RO, resourceAllocated), R(RA, infoReported)$ | Onus |
| 2 | $C(CT, SC, address \wedge status, incidentTaken)$ $C(CT, RA, incidentTaken, infoReported)$ | A_1 | $R(SC, ambReceived), R(AC, mobilityInfoSent), R(RO, resourceAllocated), R(CT, infoReported)$ | Commitment introduction <i>ii</i> |
| 3 | $C(CT, SC, address \wedge status, incidentTaken)$ $C(CT, RA, incidentTaken, infoReported)$ | $A_2 = A_1 \cup \{O(CT, infoReported)\}$ | $R(SC, ambReceived), R(AC, mobilityInfoSent), R(RO, resourceAllocated)$ | Onus |
| 4 | $C(CT, SC, address \wedge status, incidentTaken)$ $C(CT, RA, incidentTaken, infoReported)$ $C(RA, RO, infoReported, resourceAllocated)$ $C(RO, AC, resourceAllocated, mobilityInfoSent)$ $C(AC, SC, mobilityInfoSent, ambReceived)$ | A_2 | $R(AC, ambReceived), R(RO, mobilityInfoSent), R(RA, resourceAllocated)$ | Commitment introduction <i>ii</i> |
| 5 | $C(CT, SC, address \wedge status, incidentTaken)$ $C(CT, RA, incidentTaken, infoReported)$ $C(RA, RO, infoReported, resourceAllocated)$ $C(RO, AC, resourceAllocated, mobilityInfoSent)$ $C(AC, SC, mobilityInfoSent, ambReceived)$ | $A_3 = A_2 \cup \{O(AC, ambReceived), O(RO, mobilityInfoSent), O(RA, resourceAllocated)\}$ | \emptyset | Onus |

SC: service consumer, CT: call taker, RA: resource allocator, RO: radio operator, AC: ambulance crew

team) have conflicting needs, at most one of those needs can be pursued.

Conflict introduction: Introduce an assumption of a conflict into a design configuration.

$$\begin{aligned} & \langle S, A, N \cup \{R(\tau_1, p), R(\tau_2, q)\} \rangle \hookrightarrow \\ & \langle S, A \cup \{p \oplus q\}, N \setminus \{R(\tau_2, q)\} \cup \{R(\tau_1, p)\} \rangle \end{aligned}$$

If we include the above reduction, Theorem 1 fails: although consistency is preserved, we can no longer establish that the original requirements are satisfied, because some may be dropped along the way. Incorporating conflict would lead us to formalize *satisficing* [9]. Then we may seek to establish that a design process satisfies the stated requirements.

Let us see how Protos would support design in the presence of conflicts. Imagine in Table I that CAR OWNER has an additional requirement concerning data privacy, that is, $R(C, privacy)$. Conceptually, this requirement would conflict with the disclosure of personal data, that is, $R(C, disclosure)$. In Protos, the stakeholder would introduce an assumption that $disclosure \oplus privacy$ via Conflict Introduction. Then, by Conflict from Table II, we are guaranteed that one of the two propositions, i.e., one of the two requirements, cannot be met. In essence, the design process fails at this point.

V. PRELIMINARY EVALUATION

We conducted a preliminary evaluation via a modeling session involving modelers other than the authors. To this end, we recruited eight modelers, all doctoral students in computing who are familiar with goal modeling, to participate in our modeling session. At the beginning of the session, we instructed the participants in Protos concepts, reductions, and design process.

We instructed the modelers to apply Protos to jointly create a specification of Kramer and Wolf's [10] case study of the London Ambulance System (LAS), which we described for the modelers. Kramer and Wolf's LAS scenario includes nine

stakeholders, namely, the service consumer, call taker, call reviewer, LAS management, ambulance crew, call reviewer, radio operator, operator at ambulance station, dispatcher, and resource allocator. The requirement of the service consumer is to receive an ambulance when there is an incident. The call taker needs incident details to report the incident and the resource allocator requires the incident report. The radio has the requirement of the resource allocation information and the ambulance crew needs mobility instructions. For our modeling session, we merged the call taker and call reviewer stakeholders to ensure a one-to-one correspondence between the study participants and the stakeholders.

The study participants then designed the LAS STS starting from their respective requirements. Table III provides a partial design process with a few representative stakeholders where the call taker and the service consumer established a commitment and refined it prior to Step 1 of Table III. Both stakeholders take the respective onuses which are added into the assumptions set in Step 1. In Step 2, the call taker and resource allocator apply commitment introduction *ii* and the call taker commits to the resource allocator to report information about the incident upon receiving such information. In Step 3 the call taker takes the onus for this commitment. Step 4 and 5 are compressed representations of the iterative design process that is similar to Step 2 and 3, where at each step commitment introduction *ii* is applied and then the onus is taken by the respective stakeholder.

Observations. The commitment network created by the participants following Protos superficially resembles the i^* strategic dependency diagram for LAS, as provided in [11]. However, the Protos modeling of the LAS offers distinct advantages over the i^* modeling of the LAS. One, the participants were able to collectively refine the commitments to make the interaction explicit and concrete. Two, the stakeholders were able to model conditional interactions via commitments,

whereas an i^* dependency is not conditional. Three, the design reductions of Protos helped focus the design process; i^* lacks such a formalization. Four, removing the needs as the respective stakeholders took on the corresponding onuses helped the participants track the status of the design. Further, an empty needs set served as a clear stopping criterion for the design process; such a criterion does not exist in i^* .

Threats to validity. The reference document used for the LAS domains [10] includes a small number of stakeholders relative to other examples such as smart cities or a national health-care system. This somewhat narrow starting point and the participants' limited familiarity with the domain prevented them exploring designs that are not described in the reference document. Despite this, however, the participants elaborated details of interaction that are not present in the reference document by negotiating and using argumentation. The participants were collaborative and understanding to others' demands. In a real-life scenario one may expect more aggressive behavior from the stakeholders due to their conflict of interests that may even result in failure in the design process.

VI. RELEVANT LITERATURE

Our conception of an STS is in terms of a protocol specified as a set of commitments. In our conception, the STS itself, unlike Baxter and Sommerville's conception [12], does not have goals. Individuals principals, on the other hand, may have goals and may represent them in their agents.

Tropos [3] models STSs in terms of dependencies among *actors*, which can be stakeholders. Tropos dependencies differ from commitments in that they refer to the goals of the actors [13]. Tropos does not specify a protocol; instead it specifies software that implements functionality to achieve stakeholder goals. This is effectively a regimented approach. Still the idea of formal requirements analysis is one we adopt from Tropos, though we approach it quite differently: the name Protos is an anagram of Tropos. Telang and Singh [14] propose an extension of Tropos with commitments that enables specifying cross-organizational business interactions. We go further by providing the formal foundations for RE for sociotechnical systems. Dalpiaz et al. [15] illustrate how principals providing services can use commitments to and from others in reasoning about the satisfaction of their own goals. However, such work does not explain how the commitments are obtained starting from the stakeholders needs. Liu et al. [16] formalize commitments in a different sense—as a relation between a principal and a service, not between principals, as is done in Protos. Dalpiaz et al. [17] model adaptive STSs using Tropos; the approach could benefit from the adoption of Protos requirements analysis in order to obtain a well-founded and accurate specification of the STS.

Protos includes STS stakeholders explicitly in the design process and represents them explicitly in its output in contrast to RE methodologies such as KAOS [18] and problem frames [19]. Notably, neither approach supports deriving protocol specifications—like Tropos, their focus is on software specification. Mahfouz et al. [20] apply goal modeling toward

deriving choreography specifications. Choreographies specify interaction among principals but at a lower level of abstraction (via control flow constructs) than commitments.

Work on viewpoints in software engineering [21], [22] emphasizes the need to look at a system from different perspectives. In Protos, a stakeholder represents a *viewpoint*. Commitments represent a way of relating viewpoints formally (both at design and runtime). Castro et al. [23] note the complexity of Tropos models, especially the lack of modularity. Since commitments decouple principals (and their agents), they significantly alleviate the modularity problem.

Sutcliffe [24] argues for using multiple kinds of description for effective design—scenarios, design rationale, models, and so on. In the present paper, we mainly focus on deriving formal models of systems. In earlier work on commitments [25], we applied argumentation as a way of recording the design rationale of stakeholders. Extending Protos with argumentation is a direction of future work.

In mainstream software engineering, especially RE, there is increasing recognition of the importance of representing the social aspects of systems [26]. Newer approaches such as Sommerville et al.'s [27] apply *responsibility* modeling to represent contingency plans in organizations. However, reducing responsibilities to plans makes the approach a regimented one. With Protos, social and organizational factors make their appearance not only as requirements and assumptions, but in the specifications themselves as social expectations. Notice that commitments imply responsibility: a commitment for something means the debtor is responsible to the creditor for bringing about that thing. Dardenne et al.'s [18] notion of responsibility is not a relation between principals. Strens and Dobson [28] use a directed notion of responsibility as a structure that specifies not only what the responsibility-holder is responsible for but also his or her obligations. Assuming that their responsibilities are like commitments, specifying obligations in addition is unnecessary.

The need to involve end-users in the design process has been noted and addressed in the literature on Participatory Design in the fields of Human Computer Interaction (HCI) and Computer Supported Collaborative Work (CSCW) for more than twenty years (e.g., [29], [30]). Questions answered in such research range from “Who participates?”, “By what means?” and “During which part of the design process?”. Results from such research include novel design processes, such as semi-structured workshops and collaborative prototyping, new media for design, such as graphical facilitation and storyboarding, and card games, as well as cooperative prototyping and cooperative evaluation. We share many of the principles of this research. There are, however, several key differences in objectives and research methodology between our proposal and this body of research. Firstly, our focus is on design of STSs, whereas much of this literature focuses on interfaces and atomic software systems. Secondly, our focus is on the requirements problem for STSs and how to solve it through a design process, as opposed to addressing abstract user concerns. Thirdly, we are offering a formal framework for

a design process that explores a formally defined design space, amenable to formal analysis and tool support. Instead, HCI and CSCW work has emphasized the pragmatics of the design process that are outside the scope of our current research.

VII. CONCLUSIONS AND DIRECTIONS

Protos is a novel RE process for sociotechnical systems that enables refining stakeholder requirements into commitment-based system specifications. Whereas other approaches are conceptually founded upon the notion of refining requirements into machines, Protos refines them into protocols. This brings modularity to the problem space: the problem of designing principals' software (agents) is related but separate from the problem of specifying protocols. Further, it demonstrates a generalization of Zave and Jackson's foundational characterization of RE.

We emphasize the point about the divergence of the requirements of the STS stakeholders from the requirements of the principals. Supporting this divergence is the key to innovation. Simply put, whatever the goals of the stakeholders might be we simply cannot install such goals in the decision models for the individual principals.

Protos opens up some interesting and important directions for further research. First, the emphasis on multiple stakeholders suggests a deeper study of conflict at design and run time, incorporating the notion of *satisficing* requirements, possibly in relation to notions such as social welfare of the stakeholders and bringing to bear techniques such as argumentation [25] for determining which of the conflictin requirements to satisfy and which to ignore. Second, the sociotechnical setting opens up challenges of vagueness, inconsistency, and defeasibility of requirements [31]. Third, we would need to explore modeling concepts geared for requirements in diverse STS settings, e.g., with respect to the normative relationships [32], and for which we can establish results such as completeness. Fourth, it would be crucial to develop a methodology and tools that support the Protos design process.

ACKNOWLEDGMENTS

Thanks to Anup Kalia for comments on a previous version of this paper. John Mylopoulos, Paolo Giorgini, and Fatma Başak Aydemir were supported in part by European Research Council advanced grant 267856. Amit Chopra was partially supported by a Marie Curie Trentino Cofund grant. Munindar Singh was partially supported by U.S. Department of Defense under the Science of Security Lablet grant.

REFERENCES

- [1] P. Zave and M. Jackson, "Four dark corners of requirements engineering," *ACM TOSEM*, vol. 6, no. 1, pp. 1–30, 1997.
- [2] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.
- [3] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *JAA-MAS*, vol. 8, no. 3, pp. 203–236, 2004.
- [4] M. P. Singh, "An ontology for commitments in multiagent systems: Toward a unification of normative concepts," *AI & Law*, vol. 7, pp. 97–113, 1999.
- [5] S. Browne and M. Kellett, "Insurance (Motor Damage Claims) Scenario," *Doc. Identifier DI.a*, 1999.
- [6] M. P. Singh, "Semantical considerations on dialectical and practical commitments," in *Proc. AAAI*, Jul. 2008, pp. 176–181.
- [7] R. J. Back and J. Wright, *Refinement Calculus: A Systematic Introduction*. Springer-Verlag, 1998.
- [8] A. J. I. Jones and M. J. Sergot, "On the characterisation of law and computer systems: the normative systems perspectives," in *Proc. DEON*, 1993.
- [9] H. Simon, *The Sciences of the Artificial*, 3rd ed. Cambridge, MA: MIT Press, 1996.
- [10] J. Kramer and A. L. Wolf, "Proceedings of the 8th International Workshop on Software Specification and Design," *ACM SIGSOFT*, vol. 21, no. 5, pp. 21–35, 1996.
- [11] V. E. S. Souza, "An Experiment on the Development of an Adaptive System based on the LAS-CAD—Incomplete Version1," DISI, University of Trento, Tech. Rep., 2012.
- [12] G. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interact. Comput.*, vol. 23, no. 1, pp. 4–17, 2011.
- [13] A. K. Chopra and P. Giorgini, "Requirements engineering for social applications," in *Proc. i* Work.*, 2011.
- [14] P. R. Telang and M. P. Singh, "Enhancing Tropos with commitments," in *Conceptual Modeling*, Springer, 2009.
- [15] F. Dalpiaz, A. K. Chopra, P. Giorgini, and J. Mylopoulos, "Adaptation in open systems: Giving interaction its rightful place," in *Proc. ER*, 2010, pp. 31–45.
- [16] L. Liu, Q. Liu, C.-H. Chi, Z. Jin, and E. Yu, "Towards a service requirements modelling ontology based on agent knowledge and intentions," *Int. J. Agent-Oriented Softw. Eng.*, vol. 2, no. 3, pp. 324–349, 2008.
- [17] F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "Adaptive socio-technical systems: a requirements-based approach," in *Requirements Engineering*, vol. 18, no. 1, pp. 1–24, 2013.
- [18] A. Dardenne, A. Van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Sci. Comput. Program.*, vol. 20, no. 1, pp. 3–50, 1993.
- [19] M. Jackson, *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley, 2000.
- [20] A. Mahfouz, L. Barroca, R. Laney, and B. Nuseibeh, "Requirements-driven collaborative choreography customization," in *Proc. ICSOC*, 2009, pp. 144–158.
- [21] J. C. S. do Prado Leite and P. A. Freeman, "Requirements validation through viewpoint resolution," *IEEE Trans. Softw. Eng.*, vol. 17, no. 12, pp. 1253–1269, Dec. 1991.
- [22] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedcke, "Viewpoints: A framework for integrating multiple perspectives in system development," *Intl. J. Softw. Eng. Know.*, vol. 2, no. 1, pp. 31–57, 1992.
- [23] J. Castro, M. Kolp, L. Liu, and A. Perini, "Dealing with Complexity Using Conceptual Models Based on Tropos," in *Conceptual Modeling Foundations and Applications*, Springer, 2009, pp. 335–362.
- [24] A. Sutcliffe, "Integrating Design Representations for Creativity," Springer, 2013, no. 20, pp. 85–104.
- [25] A. K. Chopra and M. P. Singh, "Colaba: Collaborative design of cross-organizational business processes," in *Proc. Wkshp. Req. Eng. Syst. Services, & Syst. of Syst.* IEEE, 2011, pp. 36–43.
- [26] E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos, Eds., *Social Modeling for Requirements Engineering*. MIT Press, 2011.
- [27] I. Sommerville, R. Lock, T. Storer, and J. Dobson, "Deriving information requirements from responsibility models," in *Proc. CAiSE*, 2009, pp. 515–529.
- [28] R. Strens and J. Dobson, "How responsibility modelling leads to security requirements," in *Proc. NSPW*, 1993, pp. 143–149.
- [29] M. Muller, D. Wildman, and E. White, "Taxonomy of Participatory Design Practices," in *CACM*, vol. 36, no. 4, pp. 26–28, 1993.
- [30] L. Suchman, "Located accountabilities in technology production," in *Scand. J. Info. Sys.*, vol. 14, no. 2, pp. 91–105, 2002.
- [31] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *Proc. RE*, 2010, pp. 115–124.
- [32] M. P. Singh, "Norms as a basis for governing sociotechnical systems," in *ACM Trans. Intell. Sys. & Tech.*, vol. 5, no. 1, pp. 21:1–21:23, 2013.

Automated Detection and Resolution of Legal Cross References: Approach and a Study of Luxembourg's Legislation

Morayo Adedjouma, Mehrdad Sabetzadeh, Lionel C. Briand
SnT Centre for Security, Reliability and Trust
University of Luxembourg, Luxembourg
{morayo.adedjouma, mehrdad.sabetzadeh, lionel.briand}@uni.lu

Abstract—When elaborating compliance requirements, analysts need to follow the cross references in the underlying legal texts and consider the additional information in the cited provisions. To enable easier navigation and handling of cross references, automation is necessary for recognizing the natural language patterns used in cross reference expressions (*cross reference detection*), and for interpreting these expressions and linking them to the target provisions (*cross reference resolution*). In this paper, we propose a solution for automated detection and resolution of legal cross references. We ground our work on Luxembourg's legislative texts, both for studying the natural language patterns in cross reference expressions and for evaluating the accuracy and scalability of our solution.

Index Terms—Legal Compliance, Legal Texts, Cross References.

I. INTRODUCTION

Legal compliance is a key concern for many software applications. Information systems in domains such as government, healthcare and finance are, for example, subject to a host of laws and regulations aimed at protecting security and privacy. Systems used by governments for public administration further need to comply with public regulatory frameworks such as tax and social welfare laws. In all cases, non-compliance can have serious consequences, including fines, lawsuits, damage to public trust, loss of business, and even criminal prosecution.

To identify and elaborate compliance requirements, analysts need to review and analyze the relevant legal texts. An important complexity that arises in so doing is that legal provisions are interrelated and cannot be considered in isolation. The relationships between provisions in legal texts are captured using *cross references*. To illustrate, consider the excerpt in Fig. 1 from Article 2 of Luxembourg's Income Tax Law [1]:

Art. 2. [...] Individuals are considered non-resident taxpayers if they do not reside in Luxembourg but have a local income as per the definition of Art. 156.

Fig. 1. Using cross references for relating legal provisions

The excerpt (translated from French) has a cross reference to "Art. 156". To understand what a non-resident taxpayer is, one needs knowledge of what local income is. This example shows only one usage of cross references, i.e., specifying a dependency for a definition – in this case, the definition of local income. Cross references may be used, among other reasons, for adding exceptions and constraints, specifying priorities between provisions, and making amendments [2].

While mainly an apparatus of legal writing, cross references have implications on software requirements [2]. Maxwell et al. [3] argue that failing to consider cross references or misunderstanding their intent can lead to costly non-compliance issues in software systems. Several strands of work concerned with legal applications in the Requirements Engineering literature take cross references into consideration. For example, Breaux & Antón [4] follow cross references during requirements elaboration and analyze the cited provisions for identifying constraints, priorities, exceptions, refinements, and conflicts between compliance requirements. Ghanavati et al. [5], [6] model legal cross references as explicit goals and use these goals both for compliance analysis of business processes and for change propagation between legal requirements.

To perform the above activities in a more systematic and efficient manner, it is important to have legal texts structured as markup documents, e.g., in an XML format, with cross references represented as navigable links [7], [8]. Doing so requires the ability to first recognize the natural language expressions that denote cross references (*cross reference detection*), and then to interpret and link these expressions to the target provisions (*cross reference resolution*). The resulting links on the one hand enable easier and more structured exploration of legal texts by analysts, and on the other hand, provide a basis for further analysis, particularly traceability and impact analysis [5]. A typical legal text can contain hundreds and sometimes thousands of cross references. Automated support is thus necessary for cross reference detection and resolution.

In this paper, we develop a framework for automated detection and resolution of cross references in legal texts. Several approaches already exist for this purpose [9], [10], [11], [7], [12], [13]; but, as we argue in more detail in Section X, certain facets have not been adequately explored. Most notably:

- Cross reference detection and resolution require knowledge of the structure of legal texts, i.e., how these texts are organized into sections, articles, and so on. Typically, a schema is defined to express this structure [14], [7], [8]. Manual work is however necessary to transform a document without markup (e.g., in PDF format) into a markup document (e.g., in XML format) that conforms to the schema.
- In some countries, best practices exist on how to phrase and use cross references in legal texts. For example, in the US,

the Bluebook [15] and the US Association of Legal Writing Directors' (ALWD) Citation Manual [16] lay down specific conventions for cross references. These best practices, as already observed by others [12], are often inadequate for accurate detection of cross references, particularly in older legal corpora, e.g., public law. Grounded Theory studies of actual legal texts, e.g., as done by Breaux [7] and de Maat et al. [10], provide valuable insights about the flexible natural language patterns used for specifying cross references; however, further investigations of actual legal texts is required to understand commonalities between legal texts across different countries and to develop reusable natural language patterns for cross reference detection.

- The majority of existing work does not clearly distinguish cross reference detection and the more complex task of resolution. Consequently, important subtleties that arise during resolution have not been adequately addressed, e.g., disambiguation when the cross reference patterns are ambiguous.

To address the above, we make the following contributions:

1. We describe how a schema for the structure of legal texts can be used to *automatically* transform non-markup texts into texts with structural markup. This task is a prerequisite for cross reference resolution and, if done manually, is laborious.
2. We report on a thorough examination of the cross reference expressions in Luxembourg's Income Tax Law [1], deriving natural language patterns for these expressions that we believe are likely to generalize to other texts and other countries.
3. We provide a systematic treatment of cross reference resolution, describing the subtleties in the interpretation of complex cross reference expressions. While our treatment is applicable to both *intra-* and *inter*-document cross references, only the former category (commonly referred to as *internal* cross references) is considered in this paper.
4. Building on a Natural Language Processing (NLP) platform, GATE [17], and a first-order logic interpreter, Crocopat [18], we develop tool support for automated detection and resolution of cross references and conducting cross reference analysis.

The remainder of the paper is organized as follows: Section II provides background and terminology. Section III presents an overview of our approach, followed by Sections IV through VII, where we elaborate the technical components of the approach. Section VIII outlines tool support. Section IX describes an evaluation of our approach. Section X discusses related work and Section XI concludes the paper.

The examples used throughout the paper are from Luxembourg's Income Tax Law. This law is in French. For presentation purposes, we use English translations while preserving the structure of the original cross reference expressions.

II. BACKGROUND AND TERMINOLOGY

A (*legal*) *cross reference* is a citation that links one legal provision to another [2]. We distinguish cross references from cross reference expressions. A *cross reference expression* is a Natural Language (NL) phrase that represents one or more cross references. For example, "Articles 30 and 102 of the law

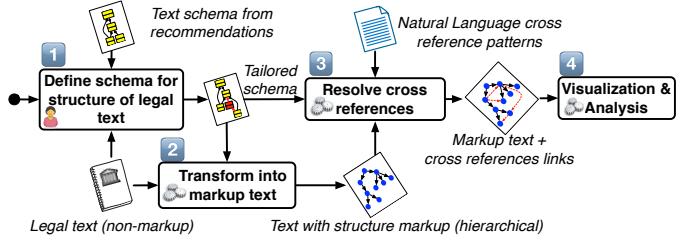


Fig. 2. Approach Overview

of April 29, 1964 concerning family benefits" is a cross reference expression. This expression embodies two cross references: one to "Article 30" and another to "Article 102" of the respective law. To avoid repetition, we abbreviate "cross reference" to **CR** and "cross reference expression" to **CRE** in the rest of the paper. We note that, with the distinction made between CR and CRE, it would be more accurate to refer to CR detection and CR resolution as CRE detection and CRE resolution, respectively. We ignore this technicality when referring to detection and resolution activities.

When a CRE points to provisions within the same legal text as where the CRE appears, the CRE is called *internal*; otherwise, when a CRE points to provisions in a different legal text, it is called *external* [19]. In the example of Fig. 1, "Art. 156" is an internal CRE. The example in the previous paragraph (i.e., "Articles 30 and 102 of ...") is an external CRE.

CREs can be further classified as explicit, implicit, or delegating. If a CRE is defined in terms of the alphanumeric labels of legal text elements, it is called *explicit*. All our examples so far where articles were referred to by their numbers were explicit. In contrast, an *implicit* CRE uses some adjective, adverb, or anaphor to refer to the target provisions [10], e.g., "this article" and "the following paragraphs". The third class, namely *delegating*, exclusively applies to external CREs. This class of CREs is used when a legislative text delegates authority to regulations for further elaboration. Regulations reside at a lower level than legislation in the hierarchy of legal texts and are usually subject to more frequent changes. Legislative texts seldom refer to regulations in a precise way and typically only indicate the nature of the regulations being cited. An example of delegating CRE is "Grand-Ducal regulation" in the following: "A Grand-Ducal regulation shall provide the details for [...]".

Finally and with regards to implicit CREs, there are occasions where legal texts use vague terms such as "provision" (in French: "disposition" or "prescription"), e.g., "the above provision". We refer to these CREs as *unspecific*. Unspecific CREs cannot be conclusively associated with specific structural elements, e.g., articles or paragraphs. They cannot thus be resolved with reasonable accuracy through automation. All CREs except for delegating and unspecific ones are in principle resolvable through automation.

III. APPROACH OVERVIEW

In this section, we provide an overview of our approach. The approach, shown in Fig. 2, has four main steps. The first step is manual and the remaining three steps are automatic.

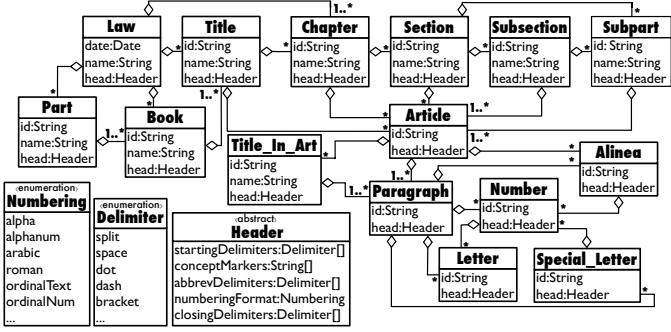


Fig. 3. Text schema for Luxembourg's Income Tax Law

Step 1 is concerned with defining a schema (metamodel) for expressing how a legal text is organized into subparts, e.g., chapters, articles, paragraphs and so on. In many jurisdictions, legal writing guidelines prescribe a generic schema for structuring legal texts. Actual legal texts may nonetheless be only partially aligned with the generic schema. Some tailoring of the generic schema may thus be necessary in order to adapt it to a specific set of legal texts. The aim of Step 1, discussed in Section IV, is to do this tailoring.

Step 2, discussed in Section V, is concerned with transforming a non-markup text (e.g., in plain text or PDF format) into a markup text (e.g., in XML format). The transformation rules are automatically derived from the schema of Step 1.

The main step of the approach is Step 3. This step, discussed in Section VI, addresses the detection and resolution of CREs. Prerequisite to this step is knowledge of the NL patterns used in the CREs. These patterns, combined with the schema from Step 1, provide the syntactic and semantic basis for the CREs. As we argued earlier, the NL patterns used in actual legal texts can be richer and more versatile than what is prescribed in best practices. This necessitates examining actual legal texts for more accurate identification of the patterns.

Step 4, discussed in Section VII, is concerned with using the outcomes of resolution for visualization and analysis.

IV. DEFINING A LEGAL TEXT SCHEMA

A text schema defines the grouping concepts, e.g., Chapter, Section and Article, used for organizing the content of a legal text. Having a text schema is essential for CR detection and resolution as both the syntax and interpretation of CREs depend on the underlying schema. We define a text schema via a UML class diagram. The classes in the diagram represent the grouping concepts used in a legal text. These classes are linked via aggregation associations representing the containment relationships between the concepts.

Fig. 3 shows the text schema for Luxembourg's Income Tax Law. To build this schema, we first developed a generic schema based on the legal writing guidelines published by the State Council of Luxembourg [20]. The resulting schema was then discussed with legal experts and enhanced with additional grouping concepts specific to the tax legislation.

An individual legal text constitutes a *Law*. A *Law* may be hierarchically organized into *Parts*, *Books*, *Titles*, *Chapters*, *Sections*, and so on. Many levels in the hierarchy are optional,

e.g., if *Part* is not used, one can go directly from *Law* to *Book*, *Title*, or *Chapter*. These alternatives are captured in the model of Fig. 3 through different aggregation paths. The core grouping concept in a Luxembourgish legal text is that of an *Article*. Articles are typically organized into *Paragraphs*. Paragraphs are often made up of *Numbers* and *Letters*. Occasionally, a special type of numbered statements, known as *Alineas*, are used for decomposing articles and paragraphs. *Title_In_Art* and *Special_Letter* are two other nuanced groups used in certain cases within articles and paragraphs, respectively.

Each grouping concept has a *header* (explained below) and an *id*. There is only one instance of *Law* per text, so the *id* is implicit in this case. Nevertheless, each law is associated with a publication date that needs to be captured. Some grouping concepts, e.g., chapters, have both an *id* and a *name* (chapter title). The header of a grouping concept *C*, called *CHeader*, provides information as to how the beginning of an instance of *C* is recognized in the text. In the model of Fig. 3, only the abstract header class is shown. Each grouping concept defines a (static) specialization of this abstract class. As an example, we show *ArticleHeader* in Fig. 4. Here, the starting delimiter is a split (carriage return or linefeed), and the concept marker is "Art" or "Article". When the abbreviated concept marker is used, it may be (optionally) followed by a dot. The numbering is alphanumeric representing the identifier of the article in the legal text. The article header closes with a dot.

| ArticleHeader |
|--|
| startingDelimiters=[split] conceptMarkers=[“Art”, “Article”] abbrevDelimiters=[dot] numberingFormat=[alphanumeric] closingDelimiters=[dot] |

(a)

Starting delimiter
concept marker
abbrev number
closing delimiter
Art. 32ter.

(b)

Fig. 4. (a) Article header class; (b) Example of an article head

V. TRANSFORMING NON-MARKUP TO MARKUP TEXT

We automatically derive from a text schema, e.g., the one in Fig. 3, regular expressions that transform non-markup legal texts to texts with structural markup. The automation builds on a simple observation: the natural structure of a textual document is such that a particular segment of text terminates only when we see a new grouping concept that is either at the same level as the current segment or at a level above the current segment. For example, assume that we have a document structured according to the schema of Fig. 3, and suppose that we are within a particular section, say Section 3. For this section to terminate, we either have to reach the beginning of Section 4, or, if there are no further sections, the beginning of a new chapter, a new title, etc.

The containment relationships between grouping concepts is never recursive, i.e., we cannot have a concept, say Chapter, which logically contains another concept, say Section, and at the same time have Sections that contain Chapters. More precisely, a text schema, when viewed as a graph, is always an *Acyclic Direct Graph (DAG)*. The transitive closure (reachability) relation in a text schema is therefore always a partial order. Let $<$ denote this partial order and let C_i and C_j be a pair of grouping concepts from the given schema. A relation $C_i < C_j$ implies that C_j directly or indirectly contains C_i . For example,

Section < Chapter, Paragraph < Section and Letter < Article are some of the relations in the partial order for the text schema of Fig. 3. Equipped with this partial order and information from the *CHeader* classes (Fig. 4), one can automatically generate the regular expressions that recognize the hierarchical structure of a legal text. Algorithm 1 shows the procedure for generating and executing these regular expressions.

Algorithm 1 Build Markup for Legal Text

- 1: Let G be a DAG whose nodes are the grouping concepts in the text schema and whose (directed) edges are the aggregations in the schema;
- 2: Let n be the number of nodes in G and let $<$ be the partial order induced by the reachability relation in G ;
- 3: For $1 \leq i \leq n$: Generate a regex HeadRegEx_i to recognize C_i headers;
- 4: For $1 \leq i \leq n$: Generate a regex SegmentRegEx_i to recognize C_i segments, i.e., a C_i header followed by any C_j header such that either $C_i \leq C_j$ or C_i and C_j are not comparable;
- 5: Run all HeadRegEx_i (in any order) on the input text;
- 6: Run all SegmentRegEx_i (in any order) on the input text;

We illustrate the regular expressions for header identification (HeadRegEx) and segmentation (SegmentRegEx) over the *Article* grouping concept. Generating the regular expression for marking the head of articles (line 3 of the algorithm) is straightforward based on the information in the *ArticleHeader* class (Fig. 4). In Fig. 5(a), we show a regular expression, named *MarkArticleHead* and written in GATE’s regular expression language [17], for marking article heads. Intuitively and in line with our previous discussion about *ArticleHeader*, *MarkArticleHead* looks for the following sequence: a split, an admissible concept marker, optionally a dot, an alphanumeric number, and a final dot. Note that at the time this regular expression is run, the text has already undergone lexical analysis with information about its tokens available.

Fig. 5(b) shows a regular expression, *MarkArticleSegment*, for marking article segments. This expression, also written in GATE’s regular expression language, recognizes and annotates the text between the head of a given article and the head of the next grouping concept instance that is not logically containable in that article. More specifically, the segment for an article starts when its head is detected. The segment ends when one of the following is detected: the head of another article, the head of a higher-level concept (as per the partial order), e.g., a chapter or a book, or the head of an unrelated concept. Since *Article* is comparable with all other grouping concepts in the schema of Fig. 3, there are no unrelated concepts to consider here. This is not the case for all concepts. For example, *Alinea* and *Special_Letter* are unrelated and should thus terminate one another’s segments if used together in the same legal text. Finally, we note that in the regular expression of Fig. 5(b), there is a special *EOD* (End Of Document) level which logically contains all grouping concepts, meaning that the occurrence of *EOD* terminates any segment at any level.

| Rule: MarkArticleHead | |
|---|--|
| (a) (((Split)+ ((Token.string=="Art") (Token.string=="Article")) ((Token.string==".")?)? (Token.kind=="alphanumeric") (Token.string==".")):ref -->:ref. <i>ArticleHead</i> = {} | |
| Rule: MarkArticleSegment | |
| (b) (((<i>ArticleHead</i>):start ((<i>LawHead</i>) (<i>PartHead</i>) (<i>BookHead</i>) (<i>TitleHead</i>) (<i>ChapterHead</i>) (<i>SectionHead</i>) (<i>SubSectionHead</i>) (<i>SubpartHead</i>) (<i>ArticleHead</i>) (<i>EOD</i>)):end):ref -->:ref. <i>ArticleSegment</i> = {} | |

Fig. 5. Example of markup rules for Article

| Line | Simple cross reference patterns |
|----------------------------------|---|
| 1 | $\langle \text{simple-ref-expr} \rangle ::= \langle \text{explicit-expr} \rangle \mid \langle \text{implicit-expr} \rangle$ |
| 2 | $\langle \text{explicit-expr} \rangle ::= \langle \text{internal-expr} \rangle \mid \langle \text{external-expr} \rangle$ |
| 3 | $\langle \text{internal-expr} \rangle ::= \langle \text{marker-term} \rangle \langle \text{num-expr} \rangle \mid \langle \text{ordinal-expr} \rangle \langle \text{marker-term} \rangle \mid \langle \text{generic-term} \rangle \langle \text{num-expr} \rangle$ |
| 4 | $\langle \text{marker-term} \rangle ::= \text{"article"} \mid \text{"articles"} \mid \text{"art."} \mid \text{"paragraph"} \mid \dots$ |
| 5 | $\langle \text{num-expr} \rangle ::= \langle \text{NUMBER} \rangle \mid \langle \text{LETTER} \rangle \mid \langle \text{ALPHANUM} \rangle$ |
| 6 | $\langle \text{ordinal-expr} \rangle ::= \langle \text{TEXT-ORDINAL} \rangle \mid \langle \text{NUM-ORDINAL} \rangle$ |
| 7 | $\langle \text{generic-term} \rangle ::= \text{"sub"} \mid \text{"under"}$ |
| 8 | $\langle \text{external-expr} \rangle ::= \langle \text{external-expr}_1 \rangle \mid \langle \text{external-expr}_2 \rangle$ |
| 9 | $\langle \text{external-expr}_1 \rangle ::= \langle \text{name-term} \rangle \mid \langle \text{category-term} \rangle \langle \text{link-term} \rangle \langle \text{DATE} \rangle \mid \langle \text{adj-term} \rangle \langle \text{category-term} \rangle \langle \text{link-term} \rangle \langle \text{DATE} \rangle \mid \langle \text{name-term} \rangle \langle \text{link-term} \rangle \langle \text{DATE} \rangle \mid \langle \text{delegating-expr} \rangle$ |
| 10 | $\langle \text{external-expr}_2 \rangle ::= \langle \text{internal-expr} \rangle \langle \text{auxiliary-term} \rangle \langle \text{external-expr} \rangle$ |
| 11 | $\langle \text{delegating-expr} \rangle ::= \langle \text{delegation-term} \rangle \mid \langle \text{adj-term} \rangle \langle \text{delegation-term} \rangle$ |
| 12 | $\langle \text{category-term} \rangle ::= \text{"law"} \mid \text{"decree"} \mid \text{"directive"} \mid \dots$ |
| 13 | $\langle \text{name-term} \rangle ::= \text{"social insurance code"} \mid \text{"complementary pension law"} \mid \dots$ |
| 14 | $\langle \text{adj-term} \rangle ::= \text{"modified"} \mid \text{"grand-ducal"} \mid \text{"ministerial"}$ |
| 15 | $\langle \text{auxiliary-term} \rangle ::= \text{"as it was introduced by the"} \mid \dots$ |
| 16 | $\langle \text{delegation-term} \rangle ::= \text{"regulation"} \mid \text{"memorial"} \mid \dots$ |
| 17 | $\langle \text{implicit-expr} \rangle ::= \langle \text{implicit-term} \rangle \langle \text{marker-term} \rangle \mid \langle \text{implicit-term} \rangle \langle \text{category-term} \rangle \mid \langle \text{marker-term} \rangle \langle \text{implicit-term} \rangle \mid \langle \text{category-term} \rangle \langle \text{implicit-term} \rangle \mid \langle \text{internal-expr} \rangle \langle \text{implicit-term} \rangle \mid \langle \text{implicit-term} \rangle \langle \text{unspecific-term} \rangle \mid \langle \text{implicit-term} \rangle \langle \text{num-expr} \rangle \langle \text{marker-term} \rangle \mid \langle \text{unspecific-term} \rangle \langle \text{implicit-term} \rangle$ |
| 18 | $\langle \text{implicit-term} \rangle ::= \text{"above"} \mid \text{"below"} \mid \text{"preceding"} \mid \text{"following"} \mid \text{"that follows"} \mid \text{"next"} \mid \text{"previous"} \mid \text{"this"} \mid \text{"in question"} \mid \text{"same"} \mid \dots$ |
| 19 | $\langle \text{unspecific-term} \rangle ::= \text{"provision"}$ |
| 20 | $\langle \text{link-term} \rangle ::= \text{"of"} \mid \text{"of the"} \mid \text{"of a"}$ |
| Complex cross reference patterns | |
| 21 | $\langle \text{complex-ref-expr} \rangle ::= \langle \text{multilayered-expr} \rangle \mid \langle \text{multilayered-expr} \rangle$ |
| 22 | $\langle \text{multilayered-expr} \rangle ::= \langle \text{multilayered-expr}_1 \rangle \mid \langle \text{multilayered-expr}_2 \rangle$ |
| 23 | $\langle \text{multilayered-expr}_1 \rangle ::= \langle \text{internal-expr} \rangle \langle \text{sep-term} \rangle \langle \text{num-expr} \rangle \mid \langle \text{external-expr} \rangle \langle \text{sep-term} \rangle \langle \text{num-expr} \rangle \langle \text{sep-term} \rangle \langle \text{DATE} \rangle$ |
| 24 | $\langle \text{multilayered-expr}_2 \rangle ::= \langle \text{multilayered-expr}_1 \rangle \langle \text{sep-term} \rangle \langle \text{num-expr} \rangle \mid \langle \text{multilayered-expr}_1 \rangle \langle \text{sep-term} \rangle \langle \text{implicit-term} \rangle$ |
| 25 | $\langle \text{multilayered-expr} \rangle ::= \langle \text{multilayered-expr} \rangle \mid \langle \text{multilayered-expr} \rangle$ |
| 26 | $\langle \text{multilayered-expr}_1 \rangle ::= \langle \text{internal-expr} \rangle \langle \text{sep-term} \rangle \langle \text{internal-expr} \rangle$ |
| 27 | $\langle \text{multilayered-expr}_2 \rangle ::= \langle \text{multilayered-expr} \rangle \langle \text{sep-term} \rangle \langle \text{internal-expr} \rangle \mid \langle \text{multilayered-expr} \rangle \langle \text{link-term} \rangle \langle \text{internal-expr} \rangle \mid \langle \text{multilayered-expr} \rangle \langle \text{link-term} \rangle \langle \text{multilayered-expr} \rangle$ |
| 28 | $\langle \text{sep-term} \rangle ::= \text{";"} \mid \text{"-"} \mid \text{"and"} \mid \text{"or"} \mid \text{"to"} \mid \dots$ |

Fig. 6. Grammar for natural language cross reference patterns

Due to space reasons, we cannot present all the technical details for generating the markup regular expressions. For example, groups such as chapters that have both ids and names have slightly more complex regular expressions than those for articles which only have ids. The annotations produced over a non-markup legal text by the regular expressions can be easily turned into a markup format, e.g., XML. The resulting markup text is the basis for the resolution process (Section VI).

VI. DETECTING AND RESOLVING CROSS REFERENCES

We automatically detect the CREs in a given legal text based on predefined patterns (Section VI-A). We then interpret the detected CREs into a set of individual CRs and link the resulting CRs to the target provisions (Section VI-B).

A. NL Patterns for Cross References Expressions

Our NL patterns were derived from a study of (Luxembourg’s) Income Tax Law [1]. The 2013 edition of this law is 189 pages long and organized into 3 titles that collectively contain 15 chapters. The law has 236 articles, composed of 767 paragraphs. The large size and complexity of this law provides a rich basis for our investigation. Basing our work

on the Income Tax Law was further motivated by our access to legal experts who could help us with understanding the structure and the content of this law. We analyzed all the 1223 CREs in the Income Tax Law and developed regular expressions to formalize the patterns observed. These patterns, shown in Fig. 6, are organized into simple and complex. Complex patterns build on top of simple patterns, providing certain advanced features discussed later in this section.

We explain and illustrate our patterns, drawing on the terminology defined in Section II. We have made minor simplifications to the patterns for better readability. While the patterns were developed over French text, they carry over to English almost verbatim. With regards to the French grammar, there is only one simplification to note: In French, ordinals can appear both before and after nouns (e.g., “paragraphe premier”, “premier paragraphe”); whereas in English, they can appear only before (e.g., “first paragraph”). In Fig. 6, symbols in upper-case letters, e.g., $\langle \text{NUMBER} \rangle$ denote terminals as identified by an NL lexical analyzer. Non-terminals that end with *term*, e.g., $\langle \text{marker-term} \rangle$ and $\langle \text{name-term} \rangle$ denote elements in predefined dictionaries (commonly known as gazetteers [17] in NLP). These terms vary from one legal jurisdiction and language to another and must be specified for a specific context.

Simple CREs. A simple CRE can be explicit or implicit (L. 1 of Fig. 6). Among explicit CREs, we distinguish internal and external (L. 2). Non-terminals $\langle \text{internal-expr} \rangle$ (L. 3) and $\langle \text{external-expr} \rangle$ (L. 8) respectively capture *explicit* internal and *explicit* external CREs.

An (explicit) internal CRE (L. 3-7) is either a concept marker, e.g., *article*, followed by a numerical expression, or an ordinal expression followed by a concept marker. The numerical expression can be an arabic number (“article 1”), a roman number (“chapter IV”), an alphanumeric (“alinea 2bis”), a number written out in text (“alinea four”), or a letter (“letter a”). A numerical expression may have brackets around it or at the end (“paragraph (2)”, “paragraph 2”). An ordinal expression can be numerical (“1st article”) or textual (“first article”).

A new pattern observed in our study is when a generic term, e.g., *under*, replaces a concept marker, e.g., *letter*. For example, “*under a*” may be used in an article instead of “*letter a*”.

An (explicit) external CRE (L. 8-16) can be as simple as just the name of an external law, e.g., “social insurance code”. Alternatively, an external CRE may be a phrase starting with an optional auxiliary term (e.g., “modified”) followed by a legal text category and a date, e.g., “modified law of 23 July 1993”. It is further possible for an external CRE to point to the internal provisions of an external law, e.g., “article 54bis as it was introduced by the Law of 23 July 1983”. Delegating references (see Section II) also fall under external CREs.

A simple CRE may be implicit (L. 17-19), e.g., “this article”. Among implicit CREs, there are some that cannot be resolved in an accurate way because they use an unspecific term, e.g., “the following provisions”. An interesting variation of implicit CREs not previously reported are those that combine implicit terms and numerical expressions, e.g., “first four alineas”.

Complex CREs. Complex CREs enhance simple CREs with three additional features: enumerations, ranges, and navigation through levels. Our classification of complex CREs follows de Maat et al.’s [10]: *multivalued* and *multilayered* (L. 21). Multilayered CREs can have multivalued parts (L. 27).

A multivalued CRE (L. 22-24) cites many provisions in the same expression by specifying only once a concept marker followed by a numerical expression. The numerical expression may be (1) an AND/OR enumeration, e.g., “numbers 1, 1a, 2 and 3” and “articles 22bis or 102”; (2) a range, e.g., “numbers 1 to 3”; or (3) a combination of enumerations and ranges, e.g., “articles 119 to 121 and 124”. Similar to simple CREs, multivalued CREs can use different numbering formats, e.g., ordinals as in “second and third alineas”. Our grammar allows the repetition of enumerations and ranges within a CRE to accommodate cases seen in our study, e.g., “articles 144, 147, 148 to 150, 158 to 160, 161, 162, and 163”. We further allow multivalued CREs to include implicit terms, e.g., “articles 26-2, 27 and the following”. Neither of these features are captured by de Maat et al. [10].

A multilayered CRE (L. 25-27) describes a navigation path through the hierarchy of a legal text. The navigation may be from an upper to a lower level, e.g., “article 91, 1st alinea, No 2”. Alternatively, the navigation can be from a lower to an upper level, e.g., “second sentence of article 10 of the law of 23 may 1964”. Finally, a navigation can be mixed-mode. That is, a CRE may start at a convenient hierarchical level, navigate upward or downward in the hierarchy, and then go in the reverse direction. For example, consider the following CREs: “article 3, paragraph (2) of the Law of 8 June 1999” and “numbers 3 and 4 of article 22bis, alinea 2”. The navigation in the former is *Article* → *Paragraph* → *Law* and in the latter *Number* → *Article* → *Alinea*. Multilayered CRE may further use multivalued CREs in their makeup, e.g., “articles 59, alinea 3, 59bis, alinea 1, 170, alineas 2 and 3, 170bis, alineas 1 et 2, 170ter, alineas 1 and 2, and 172, alineas 4 and 5”.

TABLE I
CREs IN INCOME TAX LAW

| CRE Type | # of CREs |
|-----------------------|-----------|
| Internal | 928 |
| – <i>Unspecific</i> | 45 |
| External | 295 |
| – <i>Delegating</i> | 169 |
| Explicit | 839 |
| Implicit | 384 |
| Simple | 706 |
| Complex | 517 |
| – <i>Multivalued</i> | 192 |
| – <i>Multilayered</i> | 325 |

Table I provides a breakdown of the 1223 CREs in the Income Tax Law across three different dimensions: (1) *Internal* vs. *External*: 928 CREs are internal, of which 45 are unspecific, and 295 are external, of which 169 are delegating; (2) *Explicit* vs. *Implicit*: 839 CREs are explicit and 384 are implicit; and (3) *Simple* vs. *Complex*: 709 CREs are simple and 519 are complex. Among complex CREs, 192 are multivalued and 325 are multilayered.

B. Resolving Cross Reference Expressions

The CREs that match the NL patterns go through an interpretation phase and are then linked to the cited targets.

The aim of the interpretation phase is to translate each CRE into a set of individual CRs. The main complexity arising during interpretation is that some of the NL patterns in Section VI-A are *ambiguous*, i.e., several parse trees may

exist for the same CRE. While a regular expression recognizer can delineate the start and end of each CRE even when the grammar is ambiguous, without knowing the structural markup of the underlying legal text, one cannot choose the parse tree that is suitable for the text. Parser generators such as Yacc [21] require static priorities to be defined in order to resolve ambiguities. This is inadequate for CREs, because the admissible parse tree depends on the context, i.e., the actual legal text under analysis. Custom interpretation rules are thus necessary, as we detail in this section. To remain concise in our descriptions, we assume that the legal text under analysis has been already preprocessed. In particular, we assume that (1) ordinals, roman numbers, and numbers spelled out in text have been replaced with arabic numerals; alphanumerics remain unchanged; and, (2) abbreviated concept markers (e.g., art.) have been replaced with full labels (e.g., article).

Interpreting Simple CREs.

Among simple CREs, only implicit ones and those using generic terms (e.g., sub) need treatment. We distinguish two cases for implicit CREs: (1) The ones that are semantically equivalent to *current*, *previous*, or *next* followed by a concept marker *C*: In the case of *current*, the CRE is interpreted as pointing to the segment of the same type as *C* containing the CRE (Example 1). In the case of *previous* and *next*, the CRE is interpreted as pointing to segment(s) of the same type as *C* that precede or succeed the CRE, respectively (Example 2). (2) Implicit CREs that are semantically equivalent to *same* or *this* followed by a concept marker *C*. We interpret such a CRE as being equivalent to the closest CRE of type *C* that comes before the CRE in question (Example 3). Note that in both of the cases above, our interpretation is a *best-guess* heuristic, as we do not interpret the semantics of the underlying text.

Interpreting generic terms such as *sub* needs to be done according to the conventions in the legal jurisdiction

Example 1
CRE: current article
Context: article 4 paragraph 2
Interp.: article 4

Example 2
CRE: following paragraphs
Context: article 122 paragraph 1
Interp.: paragraph 2, paragraph 2a, paragraph 3, paragraph 4

Example 3
CRE: same law
Prev. CRE: law of 8 june 1999
Interp.: law of 8 june 1999

to which the text belongs. In Luxembourg's legislation, the specific concept marker for a generic term can be inferred based on what is seen after the generic term. If the generic term is followed by a letter, the appropriate concept marker is *Letter*; otherwise, the concept marker is *Number* (Example 4).

Example 4
CRE: alinea 2, sub a
Interp.: alinea 2, letter a

Interpreting multivalued CREs. A multivalued CRE that uses an enumeration is interpreted with the concept marker added to each element of the enumeration (Example 5). For interpreting CREs involving ranges, we distinguish grouping concepts that have unique numbering across an entire legal text (e.g., *Article*) and grouping concepts (e.g., *Chapter*, *Paragraph*) whose numbering is reset when a higher-level grouping concept is seen. For the former, we browse the entire hierarchical structure of the legal text to identify the elements in the range (Example 6). For the latter (grouping concepts whose numbering is reset), the interpretation depends

on the context. We first attempt to interpret the CRE within the innermost segment in the hierarchy where the CRE appears. If the CRE cannot be interpreted meaningfully within this context, we recursively attempt to resolve the CRE in the immediate parent of the current segment and then the parent's parent and so on, until we arrive at the right level for interpreting the CRE (Example 7). It is important to emphasize that the actual elements of a range cannot be deduced independently of the legal text because alphanumerics may be used in the numbering, as illustrated in Example 6. When multivalued CREs include implicit terms, we apply the same process as for simple CREs (see Examples 1 – 4).

Interpreting multilayered CREs. For multilayered CREs that do not contain multivalued CREs, interpretation is performed by harmonizing the navigation order so that it is strictly upward or downward. For example, all the following CREs point to the same provision: (1) "article 131, 1st alinea, sub d", (2) "sub d) of article 131, 1st alinea", (3) "1st alinea, sub d) of article 131", (4) "article 131 sub d) of 1st alinea". Only (1) is in a harmonized form. The harmonized form is easy to compute based on the topological order discussed in Section V. The most complex form of CREs arises when layers are combined with ranges and enumerations. The regular expressions that detect such CREs are *ambiguous*.

To illustrate, consider the CRE in Example 8. Without knowing the structural organization of the underlying text, one cannot know if the "127

Example 5
CRE: articles 14, 61, 91 or 95
Interp.: article 14, article 61, article 91, article 95

Example 6
CRE: articles 99ter to 102
Legal Text: Lux. Income Tax Law
Interp.: article 99ter, article 99quater, article 100, article 101, article 102

Example 7
CRE: paragraphs 1 to 3
Context: article 50bis, paragraph 4
Parent context: article 50bis
Interp.: paragraph 1, paragraph 2, paragraph 3
Note: First interpretation attempt in the context of article 50bis, paragraph 4 fails. Second attempt at the level of article 50bis succeeds.

and 154ter" fragment in this CRE refers to articles, paragraphs, or numbers. One could take cues from punctuation and the singular vs. plural concept markers to rule out the fragment referring to paragraphs. One could further deduce that either 127 or 154ter has to be an article because the article concept marker is in plural form. Unfortunately, such reasoning is unreliable as punctuation and the use of singular vs. plural are not consistently followed in legal texts. For example, the distinction between singular and plural disappears when abbreviations (e.g., art.) are used.

We interpret multilayered CREs with ranges and enumerations in a similar manner to Example 7. When faced with a CRE fragment whose type is unknown, an attempt is made to interpret that fragment in the deepest context previously used for interpretation. In the case of the CRE in Example 8, this means that first, we take the numerical expression "127" (whose type is unknown) to be the continuation of "numbers 1 to 3". An attempt is made to interpret "127" in the context of "article[s] 109,1st alinea", i.e., the same context where "numbers 1 to 3"

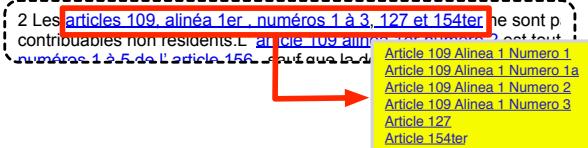


Fig. 7. A hyperlinked view of Luxembourg's Income Tax Law

was interpreted. After this attempt fails, we recursively switch to the upper-level context in the CRE, i.e., “article[s] 109” (the same context where “1st alinea” was interpreted). This second attempt also fails. The third attempt tries to interpret “127” in the context of all articles. This attempt succeeds. Now that “127” has been interpreted as an article, the CRE will be seen as if the concept marker article appeared just before “127” in the legal text. The remainder of the enumeration, i.e., “154ter”, is interpreted as if the CRE is “articles 109, 1st alinea, numbers 1 to 3, article 127 and 154ter”.

Once the interpretation phase is completed, we link each CRE to all the provisions resulting from its interpretation. The details of establishing these links are straightforward. An example of how the resulting links are rendered is provided in Section VII when discussing visualization and navigation.

VII. APPLICATIONS

In this section, we outline some important use cases that build on the results of CR detection and resolution.

Visualization and Navigation. CR detection and resolution is a prerequisite for generating navigable views of legal texts. In Fig. 7, we show a small excerpt of an HTML view of Luxembourg's Income Tax Law. In this view, resolved CREs appear as hyperlinks. Clicking on a CRE brings up a tooltip box, allowing the user to navigate to any of the CRs entailed by the CRE. A view like this is useful during the elaboration of compliance requirements, where analysts often need to follow the CRs in search of additional relevant information.

Identifying Anomalous CREs. A natural byproduct of the resolution process are diagnostics about CREs that cannot be resolved. Failure to resolve a CRE may be due to it being non-well-formed, or due to the CRE targets being non-existing. In either case, it is important for both legal experts and requirements analysts to be made aware of anomalous CREs.

Trace Link Analysis. Trace link analysis is concerned with identifying the artifacts or provisions that refer to a particular provision. While this type of analysis has a broad spectrum of applications, our work on the subject is motivated by two specific software engineering scenarios: (1) Many governmental websites and web forms, e.g., tax declaration forms, cite legal provisions to clarify how their content relates to the law. Without automated trace link analysis, it is very difficult to determine how a change in a given law affects the government's websites and forms. (2) Advanced text search: To illustrate this usage scenario, consider the following example from Luxembourg's Income Tax Law: “Art. 24” of this law elaborates the pension schemes recognized for taxation. A natural query for an analyst who is elaborating the compliance requirements for taxation of pensioners is: *Where is “Art. 24” referred to?* A naive lookup of the string “Art. 24” in the law's

```

Contains(x, y)  := Node(x) & Node(y) &
                  EX(e, Edge(e) & Type(e, "Contains") & Source(e, x) & Target(e, y));
TrContains(x, y) := TC(Contains(x,y)) | (x = y);
Cites(x, y)    := Node(x) & Node(y) &
                  EX(e, Edge(e) & Type(e, "Cites") & Source(e, x) & Target(e, y));
LinkedToArt(x, y) := Node(x) & Type(y, "Article") & EX(z, Cites(x, z) & TrContains(y, z));

```

Fig. 8. Logical queries for trace link analysis

text yields no results, despite the article being internally cited in four places. On all occasions, the article is cited within ranges: “Art. 4 to 155bis”, “Art. 14 to 108bis” and “Art. 16 to 60” (appearing twice). Without automation, identifying where a given article is being cited requires a manual scan of the entire law, i.e., 189 pages of text in the case of the Income Tax Law.

To automate trace link analysis, we need a logical representation of the structure of the legal text(s) in question along with their CR links. This information is conveniently expressed as a *typed graph* [22] – intuitively, a graph whose nodes and edges are typed. In our problem, graph nodes represent instances of the grouping concepts in a legal text, e.g., individual chapters, sections, articles, and paragraphs. Each node type is thus one of the classes of the text schema. For Luxembourg's Income Tax Law, this is the schema of Fig. 3. Edges may represent two types of relationships: first, that a grouping concept instance *contains* another, e.g. Article 4 *contains* Article 4 Paragraph 2; and second, that a grouping concept instance (directly) *cites* another, e.g. Article 4 Paragraph 2 *cites* Article 46 Number 3.

Given such a logical representation, one can infer links between any pair of grouping concept instances. For example, one can identify all citations to a given article. We show the logical queries for this computation in the snippet of Fig. 8. The snippet is written in the Relational Manipulation Language (RML) – the query language for a first-order logic interpreter, named Crocopat [18]. In the snippet, we compute a relation, Contains(x,y), that holds for all (x,y) where y is a child of x in the legal text's hierarchy tree. TrContains(x, y) computes the reachability relations via containment using the transitive closure operator (TC). TrContains(x, y) thus holds for all (x,y) where y is a descendant of x. Cites(x,y) computes (x,y) where x directly cites y via a CR. Finally, LinkedToArt(x, y) computes all (x,y) where y (i.e., the link target) is of type Article, and where x cites some element z that is transitively contained in y (e.g., a paragraph of y, or a letter in a paragraph of y).

Circularity Analysis. Cyclic citations are common in legal documents. A frequent usage is when a provision X refers to a provision Y to state that X depends on Y for a definition; and Y refers back to X to state that Y provides a definition required by X. While cycles seldom indicate errors, they need to be investigated carefully to verify absence of circular reasoning, e.g., cases where provisions X and Y both depend on one another for a definition. Circular analysis is performed using similar logical queries to what we illustrated in Fig. 8. We do not elaborate the queries due to space constraints.

VIII. TOOL SUPPORT

We implement our approach in a tool named LeCA (Legal Cross Reference Analyzer). LeCA provides automated support for (1) generation of text structure markup (Section V), (2) CR

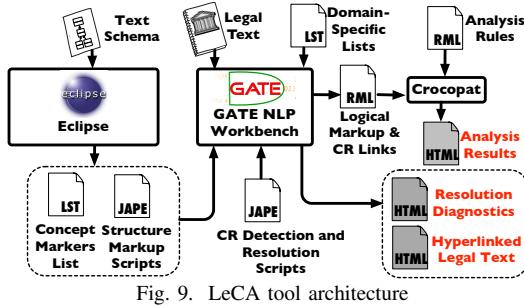


Fig. 9. LeCA tool architecture

detection and resolution (Section VI), and (3) visualization and CR analysis (Section VII). LeCA builds on the GATE Workbench [17] – a mature open-source NLP framework.

Fig. 9 shows an overview of LeCA. Eclipse’s model-to-text transformation facilities are used in order to derive, from a text schema, JAPE scripts for text structure markup. These scripts are then executed by GATE, followed by those for CR detection and resolution. The CR detection and resolution scripts rely on several keyword lists (gazetteers), e.g., concept markers, names of legal texts, and implicit terms. The list for concept markers is derived from the text schema. The remaining lists are domain-specific and depend on the language of the legal text and the specific jurisdiction to which the legal text belongs. These lists thus need to be provided by the user.

As output, LeCA produces an HTML view of the input legal text with CRs represented as hyperlinks. Diagnostics are provided for any unresolved CRs. LeCA additionally generates a logical representation of the input text’s structure and CRs, which is in turn fed to a first-order logic interpreter, Crocopat [18], for analysis. Our analysis rules currently cover traceability link generation and identification of circular citation paths.

LeCA is written in GATE’s regular expression language, JAPE (Java Annotation Patterns Engine). In total, LeCA implements 113 JAPE scripts with approximately 10,000 lines of JAPE code, excluding comments and third-party libraries.

IX. EVALUATION

In this section, we describe an evaluation of our approach, aimed at answering the following Research Questions (RQs):

RQ1. How accurate is our approach at detecting CReS? Our CRe patterns were gleaned from Luxembourg’s Income Tax Law. RQ1 aims at evaluating the completeness of the patterns by analyzing their accuracy for detecting CReS in legislative texts other than the Income Tax Law.

RQ2. How accurate is our approach at resolving CReS? RQ2 aims at measuring how accurate our approach is in resolving CReS that have been already detected.

RQ3. Is our approach scalable? Legal texts can be hundreds and sometimes thousands of pages long. RQ3 aims at establishing whether our approach runs within reasonable time.

RQ1. To answer RQ1, we selected 13 legislative texts with a total of 1640 pages from Luxembourg’s legal corpora. Our main selection criteria were to cover (1) a diverse set of texts, and (2) a large timespan in terms of when the texts were

TABLE II
RESULTS FOR RQ1

| CRE Type | # of CREs | Correctly Identified | Partially Identified | Missed |
|-----------------------|-----------|----------------------|----------------------|--------|
| Internal | 857 | 848 | 8 | 1 |
| External | 995 | 965 | 30 | 0 |
| Explicit | 1389 | 1350 | 38 | 1 |
| Implicit | 463 | 463 | 0 | 0 |
| Simple | 1031 | 1029 | 1 | 1 |
| Complex | 821 | 784 | 37 | 0 |
| – <i>Mutivalued</i> | 373 | 372 | 1 | 0 |
| – <i>Multilayered</i> | 448 | 412 | 36 | 0 |

first drafted. The selected texts were drawn from civil laws, criminal and penal laws, social welfare, pensions, housing, commerce, and healthcare. The oldest text in our selection dates back to 1808 and the newest one to 2011.

We randomly chose 10% of the pages in each selected text. If a randomly-chosen page coincided with the preface, table of contents, document history, or index, the page was discarded and another random page was considered. In total, we considered 164 pages of text containing actual legal provisions. We conducted a manual inspection of these pages and highlighted the CReS found. This inspection yielded 1852 CReS.

Following the inspection, we applied our tool for detecting the CReS in these pages. The tool was applied exclusively for detection, i.e., structural markup generation and resolution were not performed. For detection, we used the concept markers (line 4 of the patterns in Fig. 6) prescribed for legislative texts by Luxembourg’s legal writing best practices that are in effect today [20]. For generic terms, law names and auxiliary terms (respectively, lines 7, 13 and 15 of the patterns), we exploited the lists built from our investigation of the Income Tax Law. Table II summarizes the results for RQ1. In the table, we classify the identified CReS across the same dimensions as those used in Table I for the Income Tax Law.

The results indicate that our patterns miss only one CRe amongst the ones investigated (less than one tenth of a percent). This CRe was at(1) (in French, au(1)), which referred to paragraph 1 of the article in question. The CRe can be detected by adding “at” to the generic terms (line 7 of the patterns). 38 CReS ($\approx 2\%$) were only partially identified. All cases were attributed to incompleteness in the lists of concept markers, law names, and auxiliary terms. Detection further yielded 5 false positives (not shown in the table).

Without addressing the incompleteness in the lists, detection has a precision of 99.7%, recall of 97.9% and F-measure of 98.8%. If the lists are completed, these measures will respectively be: 99.7%, 100% and 99.8%. No new patterns emerged from our investigation in RQ1, providing confidence about the completeness of our patterns for Luxembourg’s legislation.

RQ2. We answer RQ2 based on the resolution of internal CReS in Luxembourg’s Income Tax Law. Although this law is the same legal text from which we drew our CRe patterns, we believe that using this text towards answering RQ2 is justified in the light of the following: First, from our analysis of RQ1, one can be reasonably confident that our patterns achieve high coverage in detection. Second, our resolution

algorithm is instantiated based on only the text schema and the CREs patterns, irrespectively of the actual legal text. We thus anticipate little bias resulting from using the Income Tax Law in RQ2. We further need to note that although our patterns address both internal and external CREs, our evaluation of RQ2 is exclusively concerned with internal ones. A detailed resolution of external CREs requires the text schema for the external legal texts.

We attempted automated resolution for all internal CREs except unspecific ones, i.e., a total of 883 CREs (see Table I in Section VI-A). This resulted in 1736 CR links plus 9 warnings for the CRs whose target provision could not be found. Two of the authors independently inspected the resulting CR links and the warnings. All the 1736 automatically-generated links were deemed correct. Of the 9 warnings, 8 arose due to CREs that were indeed anomalous in the text of law. When combined, the CR links and the 8 warnings fully covered the internal CREs over which resolution was attempted.

The last warning was due to our algorithm incorrectly interpreting an external CRE as an internal one. The CRE in question is paragraph 11bis in the following excerpt: "The modified fiscal adjustment law of 16 October 1934 is amended with the following provision, inserted into the law as paragraph 11bis: [...]" Since our approach does not analyze the semantics of sentences, it interprets paragraph 11bis as being internal although the correct treatment is paragraph 11bis of the law of 16 October 1934.

With regards to the accuracy of resolution over the Income Tax Law, we get a precision of 99.9%, recall of 100%, and *F*-measure of 99.9% when we exclude unspecific CREs. From a practical standpoint, since we rely on a human expert to resolve the unspecific CREs, it seems more reasonable to treat these CREs as false negatives. This treatment provide a better reflection of the amount of manual work saved by automated resolution. Under this treatment, we obtain a precision of 99.9%, a recall of 97.5%, and *F*-measure of 98.7%. We further note that amendments, an example of which was given above, may result in situations where our approach cannot distinguish external CREs from internal ones. We thus expect lower accuracy over laws that make a large number of amendments.

RQ3. The execution time of our approach is dominated by the resolution phase. For the Income Tax Law, interpreting CREs into individual CRs took ≈ 151 seconds. Linking the CRs to the target provisions took a further ≈ 139 seconds. The interpretation step has embedded into it the time spent for transforming the Income Tax Law from plain text to XML. The execution time associated with the identification of CREs using regular expressions was ≈ 15 seconds for the Income Tax Law and ≈ 34 seconds for the selected pages in RQ1. Execution times were measured on a laptop with a 2.3 GHz Intel CPU and 8GB of memory. Given the short overall running time of our approach, we expect it to be scalable to larger legal corpora with thousands of pages.

X. RELATED WORK

Breaux et al. [4], [7] identify natural language patterns for CREs based on a study of 118 expressions across three US

regulations. They propose the use of an explicit schema for modeling the structure of legal texts. Similar to this earlier work, we study actual legal texts for identifying CRE patterns and use an explicit model to characterize the structure of legal texts. We consider a single but complete legal text (Luxembourg's Income Tax Law) with a total of 1223 CREs. We observe new patterns, not seen in the US regulations considered. Our study covers patterns for external CREs, not considered in [4], [7]. We propose automation for text structure markup, which is a manual process in [7].

Kiyavitskaya et al. [11] automatically generate annotations for legal texts, extracting a wide range of concepts including actors, rights, obligations, and cross references. With respect to cross references, our investigation identifies new CRE patterns not reported in this earlier work. We further propose a method that automatically transforms legal text schemas into pattern matchers for generating structural markup.

The closest study to ours in terms of CRE patterns is de Maat et al.'s study of Dutch laws [10]. The differences in language aside, the patterns we observe in our investigation of Luxembourgish laws (written in French) are closely aligned with those in Dutch laws. In this sense, our study serves as a confirmatory measure for the generalizability of previously-observed patterns. In addition, we identify natural and important variations of these patterns. The main contributions of our work over de Maat et al.'s are: First, they assume that legal texts are already in a markup format with adequate structure to be transformed into the markup format required by their approach (MetaLex [8]). Our approach, in contrast, does not require pre-existing markup. Second, and more importantly, de Maat et al. do not clearly distinguish detection and resolution activities. They do not elaborate the resolution process, nor do they address the effectiveness of resolution in their evaluation. We instead provide a detailed treatment of resolution and measure its effectiveness in our evaluation.

Palmirani et al. [9] define CRE patterns based on legal writing guidelines for the Italian legal corpora. They tackle only detection and not resolution. Their approach does not address the generation of markup documents and their patterns are insufficient for recognizing many of the patterns seen in our study and that of the Dutch laws discussed above [10].

Hamdaqa et al. [12] propose an approach for resolving external CREs, whereby finite state machines are used for capturing the CRE patterns recommended by the Bluebook [15] and ALWD's Citation Manual [16]. The approach further considers markup generation through manually-written pattern matchers. The CRE patterns in this earlier work cover only external CREs and are exclusively based on best practice recommendations. Our approach provides a more thorough treatment: Our patterns encompass both internal and external CREs, and further are based on studying actual legal texts. Our approach automatically derives, from a given legal text schema, pattern matchers for text markup generation.

Tran et al. [13] apply machine learning for CR detection and resolution in Japanese legislative texts. Similar to them, we distinguish CR detection and resolution. However, our detec-

tion and resolution strategies are algorithmic and rule-based. Using machine learning can be advantageous in that it does not require an a priori specification of CRE patterns. However, Tran et al. do not consider advanced patterns with recursive structures or multiple layers similar to those identified and addressed in our study (Section VI-A). It is unknown how such patterns can be handled through learning. Further, for the patterns they do consider, they report an accuracy (F-measure) of $\approx 80\%$ and $\approx 86\%$ for detection and resolution respectively, with a combined accuracy of 67%. This, compared to rule-based techniques, is low (see Section IX).

Our work on cross reference analysis (Section VII) is close to that of Nentwich et al. [23] on consistency checking of XML documents using the xlinkit tool. xlinkit offers a rule-based language based on first-order logic and XPath (a query language for XML) for defining invariants over structured, hyperlinked documents and generation of various diagnostics. The xlinkit rule language is highly expressive and capable of capturing all the logical rules that underlie our analysis of legal cross references. xlinkit can thus be used as an alternative to the logical interpreter we use in our work (Crocopat [18]).

XI. CONCLUSION

We presented an approach for automatic detection and resolution of cross references in legal texts. Our approach complements existing work in a number of ways. In particular, the approach is parameterized by a text schema, making it possible to tailor the approach to different legal texts and jurisdictions. This text schema further allowed us to automatically enhance non-markup legal texts with the structural markup that is necessary for resolving cross references. Through a study of legislative texts in Luxembourg, we extended existing natural language patterns for cross reference expressions and provided a systematic way to interpret these expressions. We outlined the implementation of our approach in a Natural Language Processing environment. Our evaluation of the approach indicates that it is scalable and accurate in detecting and resolving cross references in Luxembourg's legislation.

Our work in this paper is part of a collaborative effort with the Government of Luxembourg on legal compliance in eGovernment services. An important challenge in this context is managing the continuous evolution of government IT systems as well as the laws and regulations that apply to these systems. Cross references provide important cues not only about the relationships between legal provisions but also about the relationships between the compliance requirements derived from these provisions. Consequently proper handling of cross references is essential for automated traceability and change analysis over compliance requirements.

In future work, we plan to conduct larger-scale evaluations of our approach and analyze the effectiveness of resolution for external cross references. We further plan to investigate how changes in legal provisions induce changes in compliance requirements. Another interesting topic for future work is to generalize our approach beyond legislative texts, and to regulations, circular letters, and parliamentary proceedings.

Being able to automatically link and navigate these different types of legal texts makes it easier to analyze the rationale behind compliance requirements and the way they need to be operationalized in software applications.

ACKNOWLEDGMENTS

Funding was provided by Luxembourg's National Centre for Information Technologies (CTIE) and Luxembourg's National Research Fund (FNR) under grant number FNR/P10/03. We are grateful to members of Luxembourg Inland Revenue Office (ACD) and CTIE, particularly, Thierry Prommenschenkel, Ludwig Balmer, Marc Blau, and Michael Masseroni for sharing their valuable knowledge and insights with us.

REFERENCES

- [1] Govt. of Luxembourg, "Modified Law of Dec. 4, 1967 (Income Tax) (Loi modifiée du 4 déc 1967 concernant l'impôt sur le revenu)," 2013.
- [2] J. Maxwell, A. Antón, P. Swire, M. Riaz, and C. McCraw, "A legal cross-references taxonomy for reasoning about compliance requirements," *REJ*, vol. 17, no. 2, pp. 99–115, 2012.
- [3] J. Maxwell, A. Antón, and J. Earp, "An empirical investigation of software engineers' ability to classify legal cross-references," in *RE*, 2013, pp. 24–31.
- [4] T. Breaux and A. Antón, "Analyzing regulatory rules for privacy and security requirements," *IEEE TSE*, vol. 34, no. 1, pp. 5–20, 2008.
- [5] S. Ghanavati, D. Amyot, A. Rifaut, and E. Dubois, "Goal-oriented compliance with multiple regulations," in *RE*, 2014, to appear.
- [6] S. Ghanavati, D. Amyot, and A. Rifaut, "Legal goal-oriented requirement language (legal GRL) for modeling regulations," in *MiSE*, 2014, pp. 1–6.
- [7] T. Breaux, "Legal requirements acquisition for the specification of legally compliant information systems," Ph.D. dissertation, North Carolina State University, 2009.
- [8] R. Hoekstra, "The metalex document server," in *10th Intl. Conf. on The Semantic Web*, 2011, pp. 128–143.
- [9] M. Palmirani, R. Brighi, and M. Massini, "Automated extraction of normative references in legal texts," in *9th Intl. Conf. on AI and Law*, 2003, pp. 105–106.
- [10] E. de Maat, R. Winkels, and T. van Engers, "Automated detection of reference structures in law," in *19th Annual Conf. on Legal Knowledge and Information Systems*, 2006, pp. 41–50.
- [11] N. Kiyavitskaya, N. Zeni, T. Breaux, A. Antón, J. Cordy, L. Mich, and J. Mylopoulos, "Automating the extraction of rights and obligations for regulatory compliance," in *ER*, 2008, pp. 154–168.
- [12] M. Hamdaqa and A. Hamou-Lhadj, "An approach based on citation analysis to support effective handling of regulatory compliance," *Future Generation Computer Systems*, vol. 27, no. 4, pp. 395–410, 2011.
- [13] O. Tran, M. Le Nguyen, and A. Shimazu, "Reference resolution in legal texts," in *14th Intl. Conf. on AI and Law*, 2013, pp. 101–110.
- [14] W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage, and R. Stevens, "Managing standards compliance," *IEEE TSE*, vol. 25, no. 6, pp. 826–851, 1999.
- [15] *The Bluebook: A Uniform System of Citation*, 19th ed. Harvard Law Review, 2010.
- [16] Association of Legal Writing Directors and D. Dickerson, *The ALWD Citation Manual*, 4th ed. Aspen Publishers, 2010.
- [17] Cunningham et al, "Developing Language Processing Components with GATE Version 7 (a User Guide)." [Online]. Available: <http://gate.ac.uk>
- [18] D. Beyer, A. Noack, and C. Lewerentz, "Efficient relational calculation for software analysis," *IEEE TSE*, vol. 31, no. 2, pp. 137–149, 2005.
- [19] A. Massey, P. Otto, and A. Antón, "Prioritizing legal requirements," in *RELAWS*, 2009, pp. 27–32.
- [20] M. Besch, "Traité de légistique formelle," 2005.
- [21] J. Levine, T. Mason, and D. Brown, *Lex & Yacc*. O'Reilly, 1992.
- [22] G. Rozenberg, Ed., *Handbook of graph grammars and computing by graph transformation (Vol. 1): Foundations*. World Scientific, 1997.
- [23] C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein, "xlinkit: a consistency checking and smart link generation service," *ACM TOIT*, vol. 2, no. 2, pp. 151–185, 2002.

Goal-Oriented Compliance with Multiple Regulations

Sepideh Ghanavati, André Rifaut, Eric Dubois

CRP Henri Tudor,

Luxembourg City, Luxembourg

{sepideh.ghanavati, andre.rifaut, eric.dubois}@tudor.lu

Daniel Amyot

EECS, University of Ottawa

Ottawa, Canada

damyot@eecs.uottawa.ca

Abstract—Most systems and business processes in organizations need to comply with more than one law or regulation. Different regulations can partially overlap (e.g., one can be more detailed than the other) or even conflict with each other. In addition, one regulation can permit an action whereas the same action in another regulation might be mandatory or forbidden. In each of these cases, an organization needs to take different strategies. This paper presents an approach to handle different situations when comparing and attempting to comply with multiple regulations as part of a goal-oriented modeling framework named **LEGAL-URN**. This framework helps organizations find suitable trade-offs and priorities when complying with multiple regulations while at the same time trying to meet their own business objectives. The approach is illustrated with a case study involving a Canadian health care organization that must comply with four laws related to privacy, quality of care, freedom of information, and care consent.

Index Terms—Legal Compliance, Multiple Regulations, Conflict Management, Goal-oriented Requirements Language

I. INTRODUCTION

With the increase in the usage of cloud-based systems, the ability to access data and software from nearly everywhere, as well as the emergence of complex networked systems (such as banks imposing their regulations on their data center service providers), organizations and software developers are bound to comply simultaneously with many more regulations from different jurisdictions and domains. Complying with multi-national regulations introduces more complexity for business analysts, software and requirements engineers. These different regulations can enforce the same rules or have some overlap (e.g., one can be more detailed than the other). They can even conflict with each other. Also, one regulation can permit one to perform an action whereas the same action in another regulation might be an obligation or an interdiction. Thus, organizations and software developers must ensure that their products and services are compliant with all relevant regulations in order to avoid the high costs associated with non-compliance (e.g., financial penalties, delays in making a service or product available, reputation loss, and even imprisonment).

In previous work [9][12], we proposed a model-based framework called **LEGAL-URN** based on the User Requirements Notation (URN) [22] for modeling legal requirements in the same notation as organizational requirements. We extended the Goal-oriented Requirement Language (GRL), which is part of URN, to capture the main objectives and the structure of legal requirements. This

extension, called Legal-GRL [11], is used to describe the goal model of the law/regulation, which is then linked to the plain GRL model of the organization (capturing its business objectives) [9]. To analyze the compliance of organizational GRL to Legal-GRL, the intentional elements in organizational GRL are linked to their corresponding elements in Legal-GRL. With the help of analysis algorithms [10], the compliance of the organization to the law/regulation is assessed. However, the framework discussed in [9] and in [12] only focuses on one regulation. In order to allow organizations to become compliant with more than one regulation, we extend the framework **LEGAL-URN** [12] to capture more than one regulation, analyze the challenges in handling multiple regulations, and identify solutions for handling them. We propose a pairwise comparison algorithm that defines the steps needed to compare two legal statements. We identify 5 cases that can occur while handling multiple regulations, and propose solutions on how to establish links between multiple regulations themselves as well as with the organizational GRL model.

In Section II, we present the related work while in Section III, we explain the **LEGAL-URN** framework together with two of its main layers. In Section IV, we define the steps needed for comparing regulations, and we provide an extension to the meta-model discussed in [11]. In Section V, we describe the pairwise comparison of each pair of statements and provide solutions for modeling them. We provide a case study from Ontario regulations in Section VI. We discuss the threats to validity and analyze the method in Section VII. Finally, we conclude our paper and explain the future work in Section VIII.

II. RELATED WORK

Handling multiple regulations, being compliant with more than one regulation at the same time, and resolving conflicts between multiple regulations are challenging activities for both researchers and organizations. In the last few decades, much work has been done to resolve conflicts in software requirements [1][7][8][28][31]. However, up to now, very little work has been done to address these issues for the legal requirements domain.

Maxwell et al. [25] are amongst the first to provide a set of techniques to help requirements engineers identify, analyze and resolve conflicts in multiple regulations. Their work identifies four patterns for internal and external cross-references. The main focus of their research is on two of the patterns, which are (i) external cross-references and (ii) internal cross-references that point to other portions of the legal text. With respect to

these patterns, a legal cross-reference taxonomy is developed with six categories. This taxonomy is used to identify the type of conflicts caused by cross-references and to provide a set of heuristics for the resolution of conflicts [31]. Although this work provides some grounds for identifying different cases when comparing multiple regulations, it mainly focuses on the conflicts caused by cross-references and not on comparing multiple regulations. This work only deals with textual requirements and not more abstract goal models of regulations.

Gordon et al. [15][16][17] introduced a framework that uses requirements watermarking and the requirements specification language (RSL) to (i) put high-level and low-level watermark standards across multiple regulations, (ii) translate the regulations to a canonical form and a set of metrics [5], and (iii) rationalize and analyze the differences and similarities between statements. In the watermarking framework, the first step is to extract and encode requirements from the two regulations with the RSL methodology. The second step is to compare the specifications, identify similarities and differences and measure the differences. Finally, the last step is to generate watermarks by identifying union disjoint and minimum watermarks. The union reconciliation aims to merge requirements from multiple jurisdictions by analyzing the dissimilarities, identifying similar requirements between the two regulations, and merging the two near-equivalent requirements into one single requirement. In a minimum watermark, the requirements from one regulation that does not exist in the other regulation are omitted while in a disjoint watermark these requirements are preserved. These steps can be repeated for the third, fourth, and n^{th} regulations to capture all requirements in a single requirements set.

Siena et al. [29] focus on extending the *Nómos* Framework [30] to capture variability in laws. This work aims to capture the “antecedents” and “consequents” of clauses in the model of a law to analyze the “applicability” and “satisfiability” of these clauses to a set of requirements and evaluate their compliance. To analyze the compliance, the approach includes situations, roles and six types of relations. With respect to the relations and the norm parts, the authors provide forward and backward reasoning for compliance.

The related work presented in this section aims to analyze legal requirements and resolve conflicts or handle multiple statements. These approaches mainly focus on the conflict cases. In our work, we focus on broader situations while dealing with more than one regulation. We also provide solutions for modeling these cases and for establishing the links between these legal models and the organization within the LEGAL-URN framework [12]. We propose solutions for compliance between multiple regulations and organizations.

On the basis of a recent literature review [13], methods are provided to ease linking legal texts and requirements with a knowledge base managing representations of the laws, such as frame-based or description logic-based formalisms, as used by artificial intelligence approaches [3][24]. This is needed for addressing the problems of completeness when selecting laws concerning specific domain, including the hierarchy between regulations used for inconsistencies management [23]. For this

latter problem, some support can be given by methods of the artificial intelligence using the legal interpretations of the laws [4]. Legal interpretation can be used as a feed for identifying different alternatives in goal and business models. In this context, LEGAL-URN can be the main method used for law modeling and analyses by domain experts and regulators. One of the strengths of our method is the tight relationship with artificial intelligence representation of knowledge about laws to reduce the gap between the specificities of legal texts and the requirements engineering models needed for analysis [3][14].

III. LEGAL-URN FRAMEWORK FOR COMPLIANCE

The LEGAL-URN framework [12] aims to support business process compliance with regulations. The LEGAL-URN has four layers for legal and organizational models:

- **Official Source Documents:** used to define the legislation on one side and organizational structures, policies and processes on the other side.
- **Hohfeldian Model:** consists of a set of Hohfeldian statements [32] together with structured elements of legal statements.
- **Legal and Organizational Goal Models:** based on URN's GRL, they capture the objectives and requirements of both organization and legislation.
- **Legal and Organizational Business Process Models:** based on URN's Use Case Maps (UCM) notation, they define the organizational business processes as well as representing steps mandated by legislation.

The LEGAL-URN framework supports five types of links between the layers of the models as well as between the legal and organizational parts: source, compliance, weighted/simple traceability, responsibility and consequence links [9].

This framework aims to model regulations in the same notation as organizational goals, objectives and business processes. However, some aspects of regulations are not easily captured by GRL or UCM. Thus, URN has been extended with a *Legal profile* to capture legal elements. The new extensions are called *Legal-GRL* and *Legal-UCM*.

To formalize the mapping between regulations and Legal-GRL, we introduce the Hohfeldian model layer, based on frames to capture the deontic and ontological aspects of legal texts. Here, we briefly discuss Hohfeldian and Legal-GRL models since the compliance analysis is done on the GRL level. More details about the framework are discussed in [12].

A. Hohfeldian Model of Regulations

The Hohfeldian model, which is an intermediate level between the legal text and the Legal-GRL model, uses Hohfeldian concepts for classifying legal statements, as well as legal statement structural elements such as *subject*, *modality*, *verb*, *actions*, *preconditions*, *exceptions* and *cross-references* (see the Hohfeldian model structure on the left part of the Figure 1 and in Figure 2). Like most frame-based methods that extract knowledge from texts, we identified the following rules:

- **Rule 1** – Each legal statement shall be atomic. This means that each legal statement contains one legal «actor» (the subject) and one «modal verb» (modality). The statement

- can also have one to many legal «clause» («verb» and «actions»), 0 to many legal «cross-reference», 0 to many «precondition» and 0 to many «exception».
- **Rule 2** – If a legal statement contains more than one modal verb, it must be broken down into atomic statements.
 - **Rule 3** – Legal exceptions must be also broken down into separate atomic legal (exception) statements.
 - **Rule 4** – Preconditions must be linked to either a clause or an exception.
 - **Rule 5** – When a precondition (P) for the clause (C) holds for the legal actor (A), the clause is obligatory (Ob). When a precondition (P') for the exception (Ex) holds, the exception is obligatory (Ob).
 - **Rule 6** – If there is an internal cross-reference, we replace the referencing part with the referenced statement and break the statement into atomic statements. External cross-references also break into atomic statements but they are mapped to the original legal statement via links.

Each Hohfeldian statement, based on Hohfeld's definitions, can be one of the following types: Duty-Claim, Privilege-Nonclaim, Immunity-Disability, or Power-Liability. These statements are refined to permissions or obligations in the Legal-GRL level. The semantics of our Hohfeldian model is described with the GRL evaluation model [12] and with the help of dependency links. When a precondition in the Legal-GRL model is not triggered, it is annotated with a «no-precondition» tag and is removed from the analysis. Without considering the GRL evaluation method, this roughly corresponds to the following logic formulas:

$$((P \wedge \neg P') \Rightarrow C_{ob})$$

and

$$((\neg P \wedge P') \Rightarrow Ex_{ob}).$$

B. Legal-GRL Modeling for Regulations

In [11], we explain how to create a Legal-GRL model of regulations. For this, Hohfeldian statements are refined to *permission* and *obligation* (soft-) goals. To capture preconditions, exception and external cross-references, we introduce three other types of goals, also mapped to the Hohfeldian model: precondition goals/softgoals, exception goals/softgoals and XRef goals/softgoals. Figure 1 presents the mapping between the Hohfeldian and the Legal-GRL models.

| HOHFELDIAN MODEL | LEGAL-GRL MODEL |
|------------------|-----------------------------------|
| SECTION | - |
| SUBJECT | ACTOR, EXPECTIONACTOR |
| MODAL VERB | OBLIGATION, PERMISSION STEREOTYPE |
| CLAUSE | INTENTIONAL ELEMENT |
| PRECONDITON | PRECONDITION INTENTIONAL ELEMENT |
| EXCEPTION | EXCEPTION INTENTIONAL ELEMENT |
| XREF | CROSSREFERENCE IE |

Figure 1 – Mapping between Hohfeldian and Legal-GRL Models

IV. COMPARISON ANALYSIS OF REGULATIONS

Comparing multiple regulations implies comparing pairs of statements from different regulations to determine whether they are independent, similar, complementary, or contradictory.

However, as each regulation can contain many statements, trying to analyze all possible pairs is not practical. This scalability issue can be mitigated by taking advantage of the structure of regulations. The legal text schema defined in [1], states that a law can have *books*, *chapters*, *parts*, *titles* and/or *sections*. Indeed, related statements are often clustered under one of these categories or various levels of sub-sections or sub-parts. Chapters and books can also contain sections. Parts can be used as an alternative to sections. By following these structures, it is possible to compare two legal statements which are categorized under the same book titles, chapters, sections or parts. In this paper, we focus on comparison based on section; however, if the regulation does not include a section, parts, chapters or books can be used alternatively.

By matching pairs of related sections (from two regulations) first, we can focus on pairs of statements coming from each section, and prune out the pairs that involve statements of a matched section and statements of unmatched sections. Such section-focused pairwise analysis can hence reduce the number of pairs to compare drastically.

To capture the *section* concept, we extend the Hohfeldian meta-model of [11]. Figure 2 shows the extended meta-model, which now includes one new meta-class (Section, with a name attribute). Note that, Section can be replaced by part, book, chapter, etc. Each Hohfeldian model is composed of 1 to many sections, and refers to possibly many statements. Each section can have 0 to many subsections and each section or subsection can have 0 to many Hohfeldian statements. The rest of the Hohfeldian meta-model remains the same.

Figure 3 presents an excerpt of a Hohfeldian model for Article 39(2) of the Freedom of Information and Protection of Privacy Act (FIPPA) [27]: *Notice to individual- Where personal information is collected on behalf of an institution, the head shall, unless notice is waived by the responsible minister, inform the individual to whom the information relates [...]*

The pairwise comparison algorithm, shown in Figure 5 can then take advantage of this new data structure to classify matching pairs of regulation statements. Finding sections of a regulation that are relevant for the sections of another regulation requires manual effort from experts (e.g., lawyers). However, this can be done once for each pair of laws/regulations, so this effort can be amortized when this information is reused by multiple organizations. There are five cases considered in *compareStatements()*, and they are explained in Section V.

The comparison between two legal statements is done at the Hohfeldian model layer, i.e., two actors, two modal verbs, two clauses, etc. are compared to each other based on their natural language attributes. In the pairwise comparison, the synonyms of the natural language text of each element are considered by the expert (e.g., Lawyer).

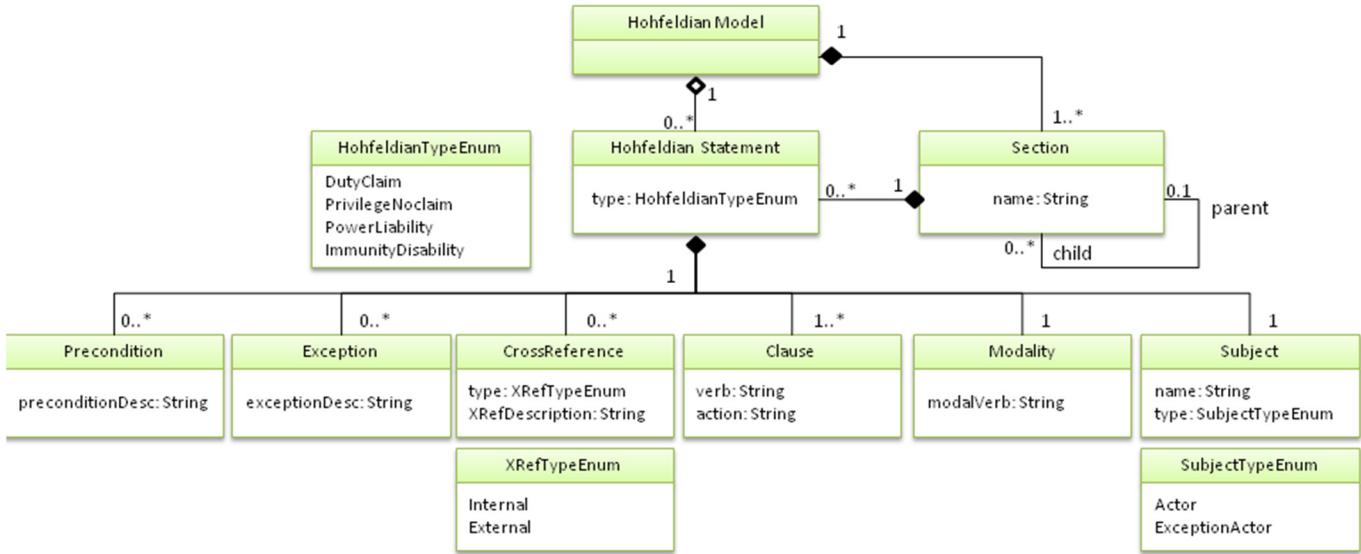


Figure 2 – Hohfeldian Meta-Model

| | |
|--------------------|--|
| SECTION | Notice to individual |
| ARTICLE# | FIPPA-39(2) |
| ACTOR | Head of institution |
| MODAL VERB | Shall |
| CLAUSE | Inform individual [...] |
| PRECONDITON | Where PI is collected on behalf of [...] |
| EXCEPTION | Notice is waived by responsible minister |
| XREF | - |

Figure 3 – Hohfeldian Model of FIPPA 39(2)

To create the Legal-GRL model for multiple regulations we follow the steps below:

- Step 1. Identify relevant legal and organizational documents.
- Step 2. Develop a Hohfeldian model for each of the regulations identified in Step 1 by classifying each statement of the legal document based on Hohfeld's classes of rights [32] and identifying the Hohfeldian elements (i.e. actor, modal verb, clause, precondition, exception and cross-reference) defined in the Hohfeldian model layer, while linking them to the source legal document (via source links).
- Step 3. Develop the goal model of the law for each of the regulations and annotate the intentional elements with «Permission», «Obligation», «Precondition», «Exception», and «XRef» tags. Create source links to the legal documents, and compliance links to the Hohfeldian model.
- Step 4. Use the pairwise comparison algorithm (Figure 5) to classify the various cases, and establish traceability links between the legal models for cases 2 to 4 (see Section V). The traceability links between low-level tasks of the two models are *weighted* traceability links with contribution factors at 100 (on a scale that goes from -100 to +100). The *weighted* traceability links ensure that the satisfaction

values of one legal model propagate to the other legal models connected to it. Between matching pairs of high-level goals or actors from the two legal models we find *simple* traceability links (correlation links with contribution factors at 0).

- Step 5. Develop the GRL model of organization and establish traceability links between Legal-GRL and GRL.

V. PAIRWISE COMPARISON OF TWO STATEMENTS

When dealing with more than one set of requirements documents and in our case more than one regulation (Step 4 in the previous section), the following cases involving two statements can be observed:

- Case 1 - There is nothing in common between the two statements.
- Case 2 - Both statements are similar to each other.
- Case 3 - One statement is complementary to the other statement.
- Case 3' - One statement is a subset of the other statement.
- Case 4 - One statement is stricter than the other statement.
- Case 5 – One statement contradicts the other statement.

Based on the rule 1 in Section III, we define the structure shown in Figure 4 for Statement_i and Statement_j.

The function compareStatement (Statement_i: HohfeldianStatement, Statement_j: HohfeldianStatement) from the algorithm shown in Figure 5, where Statement_i comes from the first Hohfeldian model (hm1) and Statement_j from the second model (hm2), is what is being defined in this section.

Each of the cases is discussed here. Note that «section»s of the statements are not compared as they have already been found to match by an *expert legal modeler*. In addition, cases 1, 2, and 5 are symmetric, but cases 3, 3', and 4 are asymmetric and hence need to be checked in both directions.

| Statement _i | Statement _j |
|------------------------|------------------------|
| A _i | A _j |
| MV _i | MV _j |
| ∀i C _i | ∀j C _j |
| ∃i P _i | ∃j P _j |
| ∃i Ex _i | ∃j Ex _j |
| ∃i XR _i | ∃j XR _j |

Figure 4 – Anatomy of Statement_i and Statement_j

A. Case 1 - Nothing in Common between the Two Statements

Statement_i is dealing with an issue different from Statement_j.

In this case, at least «actor» (i.e., a «subject» of type Actor) and/or the «clause» («verb» and «action») and/or «exception» parts of each statement are different from each other. «Precondition» leads to a «clause» or an «exception». They can be different or not and still case 1 will hold. If the «actor» are similar or one is the subset of the other actor, but the «clause» and/or «exception» are different then still case 1 is hold. However, «modal verb» and «XRef» are not necessarily different. Two statements can be directed to the same actor, have the same type of modality and be related to the same cross-referenced statement but have different concerns. Table 1, case 1 provides the summary of the pairwise comparison for this case.

$$\forall i, j: ((A_i \cap A_j) = \emptyset) \vee ((C_i \cap C_j) = \emptyset) \vee \\ (((A_i \cap A_j) = \emptyset) \vee ((C_i \cap C_j) = \emptyset) \wedge (Ex_i \cap Ex_j) \\ = \emptyset)$$

In this case, to ensure the compliance with both of the regulations, it is necessary to model Statement_j of hm2 as is, add it to the Legal-GRL model, and provide the necessary links to the organizational GRL model (Step 5). There is no link between the two Legal-GRL models.

For example, Article 22 of PHIPA [20] and Article 47 of the Health Care Consent Act (HCCA) [19] both deal with the concept of “incapacity”.

Article 22 of PHIPA states: *Determination of incapacity - An HIC that determines the incapacity of an individual to consent to [...] of personal health information [...] shall do so in accordance with the requirements and restrictions, [...]*

Article 47 of HCCA states: *Incapacity - An evaluator shall, [...], provide to persons found by the evaluator to be incapable wrt admission to a care facility such information about the consequences of the findings as is specified in the guidelines.*

Article 22 of PHIPA discusses the “incapacity of an individual to consent” and the responsibility of the HIC with respect to it while Article 47 of HCCA talks about an “incapacity for treatment”. The «clause» and «precondition» of these two articles are dealing with two different issues. To be compliant with both regulations, organization needs to have both statements included in the Legal-GRL model.

B. Case 2 - Both Statements are Similar to Each Other

Statement_j of contains «actor» («subject»), «modal verb», «clause» («verb» and «action»), «precondition», «exception» and «XRef» similar to those of Statement_i. In short, Statement_i ≡ Statement_j.

In this case, compliance with one statement ensures compliance with the other statement. However, to avoid any potential non-compliance in the face of change, it is necessary to model both of the statements in Legal-GRL and use traceability links between the two Legal-GRL models. The satisfaction values from one Legal-GRL model is propagated to the other Legal-GRL model via traceability links.

For instance, Article 25, Transfer of request, in the Freedom of Information and Protection of Privacy (FIPPA) [27] and Article 18, Transfer of request, in Municipal Freedom of Information and Protection of Privacy Act (MFIPPA) [26] both state: *Where an institution receives a request for access to a record and the head considers that another institution has a greater interest in the record, the head may transfer the request and, if necessary, the record to the other institution, within fifteen days after the request is received, in which case the head transferring the request shall give written notice of the transfer to the person who made the request.*

As illustrated in Figure 6, these two articles are addressing the same issue and have the same actors, preconditions, modal verbs, and clauses (without any exceptions or cross-references). Therefore, having a link between them (to manage change) and a link between one of them and the organization model will be enough for ensuring the compliance.

C. Case 3- One Statement is Complementary to the Other Statement.

Statement_j and Statement_i have their «actor» in common or complementary. Both statements have at least one «clause», 0 to many «precondition»(s), «exception»(s) or «XRef»(s) in common but each statement has at least one «clause» and 0 to many «precondition»(s), «exception»(s) or «XRef»(s) in addition to the common parts. «Modal Verb» for both statements are not necessarily similar but they are also NOT contradicting with each other. Table 1 formalizes this case.

$$Statement_i \cup Statement_j = Statement_i + Statement_j - (Statement_i \cap Statement_j)$$

In this case, we model both statements in Legal-GRL models but only link one of the statements as well as the complementary part of the second statement to the organizational model (Step 5). Again, the part of the Legal-GRL models that are similar are connected to each other by traceability links.

Article 44 from FIPPA states: *Personal information banks - A head shall cause to be included in a personal information bank all personal information under the control of the institution that is organized or intended to be retrieved by the individual's name or by an identifying number, symbol or other particular assigned to the individual.*

Article 10 of The Privacy Act [6] states: *Personal information banks - The head of a government institution shall cause to be included in personal information banks all*

```

Algorithm PairWiseComparison
Input hohfeldianModels: list of HohfeldianModel
Output pairedStatements: set of <Integer, HohfeldianStatement, HohfeldianStatement>

hm1, hm2: HohfeldianModel
sectionSet: set of Section
sectionHM1, sectionHM2: Section
statementHM1, statementHM2: HohfeldianStatement
index1, index2: Integer
                                         // two models to compare
                                         // set of sections
                                         // two sections to compare
                                         // two statements to compare
                                         // two indices

// compare all unique pairs of models
pairedStatements = ∅
index1 = 0
while index1 < hohfeldianModels.size()
{
    hm1 = hohfeldianModels.at(index1)
    index2 = index1 + 1 // avoids comparing previously checked pairs
    while index2 < hohfeldianModels.size()
    {
        for each sectionHM1 in hm1
        {
            // get, manually, the set of sections in hm2 that relate to sectionHM1
            sectionSet = sectionHM1.manualSectionMatchingIn(hm2)
            // compare the statements for pairs of relevant sections
            for each sectionHM2 in sectionSet
                for each statementHM1 in sectionHM1
                    for each statementHM2 in sectionHM2
                    {
                        // determine case (1 to 6) and document it.
                        case = compareStatements(statementHM1, statementHM2)
                        pairedStatements.add(case, statementHM1, statementHM2)
                    }
                }
            }
        }
    }

return pairedStatements

```

Figure 5 – Pairwise Comparison Algorithm

personal information under the control of the government institution that (a) has been used, is being used or is available for use for an administrative purpose; or (b) is organized or intended to be retrieved by the name of an individual or by an identifying number, symbol or other particular assigned to an individual.

Figure 7 presents the comparison between the two articles. Article 10(1-2) includes the same «actor» and «clause» as those in Article 44. However, Article 10(1-2) contains additional «precondition» and «exception» that Article 44 does not entail.

This article also includes an exception rule: Exception for *Library and Archives of Canada – (2) Subsection (1) does not apply in respect of personal information under the custody or control of the Library and Archives of Canada that has been transferred there by a government institution for historical or archival purposes.*

D. Case 3'- One Statement is a Subset of the Other Statement.

This case is the subset of case 3. In case 3, both statements could have additional clauses, precondition, exception or XRef, while in this case, one of them has an additional clause, precondition, exception or xref.

$$(Statement_i \cup Statement_j = Statement_j) \\ \text{or } (Statement_i \cup Statement_j = Statement_i)$$

Statement_j includes «actor» similar Statement_i or one «actor» is the generalization class of the other actor. In addition, Statement_j has minimum the same «clause», «precondition», «exception» and «XRef» of Statement_i, with some additional «clause» and potentially additional «precondition», «exception» or «XRef». This means Statement_j includes further rules. For example, Statement_i could deal with the disclosure of PHI to hospital researchers whereas Statement_j would deal with disclosure of PHI to hospital researchers as well as external researchers. The formalization in Table 1 is asymmetric; therefore the same verification must be done by swapping Statement_j and Statement_i. «Modal Verb» attributes for both statements are not necessarily similar but they are also *not* contradicting each other.

Similar to the previous case, compliance with the superset statement is enough, though both statements need to be modeled and linked through traceability links.

Article 29 in PHIPA [19] states: *Requirement for consent - An HIC shall not collect, use or disclose PHI about an individual unless, (a) it has the individual's consent and the collection, use or disclosure, as the case may be, to the best of the custodian's knowledge, is necessary for a lawful purpose; or (b) the collection, use or disclosure, as the case may be, is permitted or required by this Act.*

| HM | FIPPA -25 | MFIPPA - 18 |
|----|---|---|
| S | Transfer of Request | Transfer of Request |
| A | The Head | The Head |
| MV | May | May |
| C | Transfer the request[...] within 15 days after received | Transfer the request[...] within 15 days after received |
| P | Where institution receives | If institution receives |
| EX | - | - |
| XR | - | - |

Figure 6 – Case 2 – Pairwise Comparison Example

| HM | FIPPA -44 | PRIVACY ACT - 10 |
|----|---|---|
| S | Personal information banks | Personal information bank |
| A | A Head | The Head of the government |
| MV | Shall | Shall |
| C | Caused to be included in a PI bank all PI under control | Caused to be included in a PI bank all PI under control |
| P1 | Is organized or intended to [...] | Is organized or intended to [...] |
| P2 | - | Has been used, [...] |
| EX | - | Subsection 1 does not [...] |
| XR | - | - |

Figure 7 – Case 3 – Pairwise Comparison Example

Article 41 (1) of FIPPA mentions: *An institution shall not use personal information in its custody or under its control except, an educational institution may use personal information in its alumni records and a hospital may use personal information in its records for the purpose of its own fundraising activities, if the personal information is reasonably necessary for the fundraising activities.*

Figure 8 shows the result of the comparison. The clause, exception and precondition of article 29 of PHIPA covers more detail than article 41 of FIPPA. Compliance with PHIPA leads to compliance with FIPPA too. However, to manage the potential changes, both of the statements are modeled in Legal-GRL and the links between them are established.

E. Case 4- One Statement is Stricter than the Other Statement.

Statement_j provides stricter modality or clause than Statement_i. In this case, «modal verb» in Statement_j indicates an obligation while in Statement_i it indicates a permission, or the «clause» in Statement_j is stricter (e.g., in terms of time) than the «clause» in Statement_i. The other parts in both statements remain similar (see Table 1). Note again that the same verification must be done by swapping Statement_j and Statement_i. Note that, only one of the conditions for «modal verb» or «clause» needs to be satisfied.

In this case, it is only necessary to be compliant with the stricter statement, i.e., being compliant with Statement_j implies compliance with Statement_i. However, it is also up to the organization to decide to be compliant with which regulation. If an organization decides to be compliant with the less strict statement, it will have the non-compliance consequences which are handled in the LEGAL-URN framework in [12].

Similar to case 2, to be able to manage changes, we model both of the statements in Legal-GRL and link the intentional elements and actors of both models to each other via traceability links.

Article 47(2) - Right of Correction of FIPPA states: *Every individual who is given access under subsection (1) to personal information is entitled to,(c) require that any person or body to whom the personal information has been disclosed within the year before the time a correction is requested or a statement of disagreement is required be notified of the correction or statement of disagreement.*

Article 12(2) of the Privacy Act [6] mentions: *Every individual who is given access under paragraph (1)(a) to personal information that has been used, is being used or is available for use for an administrative purpose is entitled to (c) require that any person or body to whom that information has been disclosed for use for an administrative purpose within two years prior to the time a correction is requested or a notation is required under this subsection in respect of that information (i) be notified of the correction or notation, and [...].*

These two articles are talking about the right of correction under the same condition. However, Article 12(2) gives more time for correction than Article 47(2) (hence, the latter is stricter than the former).

| HM | FIPPA -41 | PHIPA- 29 |
|----|--------------------------|--|
| S | Use Personal information | Personal information bank |
| A | An Institution | An HIC |
| MV | Shall | Shall |
| C | Not used PI [...] | [Not] collect, use or disclose [...] |
| EX | May use | May collect, use or disclose |
| P1 | It is necessary | Has consent and is necessary |
| P2 | - | Collection, use, disclosure is permitted |
| XR | - | - |

Figure 8 – Case 3' – Pairwise Comparison Example

F. Case 5- One Statement Contradicts the Other Statement.

Statement_j is in conflict with Statement_i when both statements have a common «actor» and an actor is the generalized form of the other actor and «modal verb» and «clause» of one statement is in contradiction with the «modal verb» and «clause» of the other statement. If «precondition»s or «exception»s of the statements are contradicting with each other, there will be a conflict case. «XRef» can also be contradictory but since the «XRef» statements are themselves, atomic statements, the similar pairwise comparison can happen between them too.

In this case, complying with the first statement results in non-compliance with the second statement, and vice versa. To resolve the conflict however, it is necessary to ask a subject matter expert (e.g., a lawyer, a legal consultant, or a policy analyst), and incorporate the solution into the Legal-GRL.

Article 12(1) of the Privacy Act states: *Right of access - every individual who is a Canadian citizen or a permanent resident within the meaning of [...] has a right to and shall, on request, be given access to (a) any personal information about the individual contained in a personal information bank; and (b) any other personal information about the individual under the control of a government institution wrt which the individual*

Table 1- Summary of the Pairwise Comparison

| Statements | Case 1 | Case 2 | Case 3/3' | Case 4 | Case 5 |
|-------------------------|---------------------------------------|----------------------------------|--|--------------------------------------|---------------------------------|
| Actor (A) | $\exists i, j: A_i \cap A_j = \phi$ | $A_i \equiv A_j$ | $(A_i \equiv A_j) \vee (A_i \in A_j)$ | $A_i \equiv A_j$ | $A_i \equiv A_j$ |
| Modal Verb (MV) | - | $MV_i \equiv MV_j$ | - | $(MV_i \in Pr) \wedge (MV_i \in Ob)$ | $MV_i \zeta MV_j$ |
| Clause (C) | $\forall i, j: C_i \cap C_j = \phi$ | $\forall i, j: C_i \equiv C_j$ | $\forall i, j: (C_i \subseteq C_j) \vee (C_j \subseteq C_i)$ | $\forall i, j: C_i \Rightarrow C_j$ | $\forall i, j: C_i \zeta C_j$ |
| Precondition (P) | $\exists i, j: P_i \cap P_j = \phi$ | $\exists i, j: P_i \equiv P_j$ | $\exists i, j: (P_i \subseteq P_j) \vee (P_j \subseteq P_i)$ | $\exists i, j: P_i \equiv P_j$ | $\exists i, j: P_i \zeta P_j$ |
| Exception (Ex) | $\exists i, j: Ex_i \cap Ex_j = \phi$ | $\exists i, j: Ex_i \equiv Ex_j$ | $\exists i, j: (Ex_i \subseteq Ex_j) \vee (Ex_j \subseteq Ex_i)$ | $\exists i, j: Ex_i \equiv Ex_j$ | $\exists i, j: Ex_i \zeta Ex_j$ |
| XRef (XR) | - | $\exists i, j: XR_i \equiv XR_j$ | $\exists i, j: (XR_i \subseteq XR_j) \vee (XR_j \subseteq XR_i)$ | $\exists i, j: XR_i \equiv XR_j$ | $\exists i, j: XR_i \zeta XR_j$ |

is able to provide sufficiently specific information on the location of the information as to render it reasonably retrievable by the government institution.

Article 9(1) of the Personal Information Protection and Electronic Documents Act (PIPEDA) [18] states: *When access prohibited - An organization shall not give an individual access to personal information if doing so would likely reveal personal information about a third party.*

Article 12(1) obliges organizations to give access to an individual (even if it has a precondition about the revealing of personal information about a third party) while Article 9(1) obliges organizations *not* to give access to an individual. These two articles are in contradiction with each other. To be able to resolve this conflict, it is necessary to get advice from a legal expert in order to decide whether the access to the personal information reveals the conflict or not. Based on the result of the consultation, it is possible to decide which one of these two Articles applies to the organization.

VI. CASE STUDY

In our case study, we select four regulations: a) PHIPA, b) Quality of Care Information Protection Act, 2004 (QCIPA) [21], c) FIPPA and d) HCCA and six organizational business processes. From each of the regulations, few statements are chosen. Table 2 shows the number of statements selected from each of the regulations.

Table 2 – Number of Statements Selected

| PHIPA | QCIPA | FIPPA | HCCA |
|-------|-------|-------|------|
| 11 | 3 | 4 | 2 |

We perform the pairwise comparison between PHIPA (our base regulation) and the three other regulations and identify the

cases. The result of the comparison is shown in Table 3. As seen from the table, case 1 has the highest number in all of the three comparisons. If the pairwise comparison algorithm starts from finding and eliminating the conditions that case 1 applies, then the number of comparison for the rest of the regulations will decrease. This will help reducing the effort of the comparison. The result of the pairwise comparison was evaluated by two other researchers as well.

After the comparison is done, the Legal-GRL model for each of the legal statements is created. In case 1, the Legal-GRL models do not have any link to each other. In case 2 to 4, we connect parts of the Legal-GRL models which are in common between the two models via *traceability* links. In case 5, a subject matter expert such as a lawyer should intervene. In our case study, we identified that the QCIPA states in Article 2 that in the case of a conflict with other acts or regulations, this act prevails. Thus, for the three conflicting cases, we modeled QCIPA and linked them to the organizational GRL models.

Figure 9 presents the link between Article 41(1) of FIPPA with Article 29 of PHIPA in Legal-GRL. The pairwise comparison between the two article matches case 3' (Figure 8).

Table 3 – Pairwise Comparison of PHIPA & 3 Regulations

| Regulation | C.1 | C.2 | C.3 | C.4 | C.5 |
|--------------|-----|-----|-----|-----|-----|
| QCIPA | 22 | 1 | 7 | - | 3 |
| FIPPA | 37 | - | 7 | - | - |
| HCCA | 21 | 1 | - | - | - |

In this case, the high-level goals of the two models are linked through *simple* traceability links and the low-level intentional elements in FIPPA are linked to the low-level intentional elements in PHIPA. As seen in this figure, PHIPA has more intentional elements than FIPPA, which was also identified through case 3'.

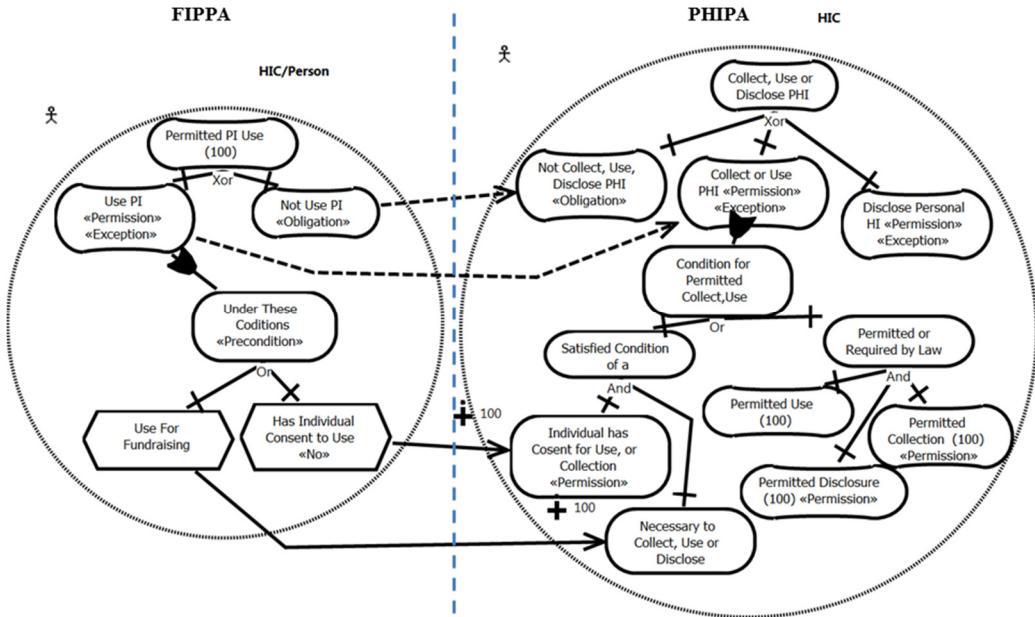


Figure 9 – Links between Legal-GRL Models of FIPPA and PHIPA

VII. THREATS TO VALIDITY

In our study, we evaluated our algorithm for pairwise comparison by incremental construction of the case study. To mitigate the threats to construct validity, we first started by comparing the QCIPA with PHIPA, and defined the 6 pairwise comparison cases. Next, we added FIPPA to the model and compared it to PHIPA. Finally, the addition of the HCCA was the final demonstration of the appropriateness of the approach. We eliminated the pairwise comparison between QCIPA-FIPPA, QCIPA-HCCA and FIPPA-HCCA to reduce the size of the case study. This elimination can be a threat to the validity of our compliance analysis result. We used our case studies from a real hospital in Ontario Canada, but, we never implemented the recommendations in the hospital. However, the models and the pairwise comparison analysis methods were reviewed and confirmed by two reviewers, one of whom is a URN expert. To mitigate internal validity threats, we aim to, first, semi-automate our pairwise comparison process through Cosine Similarity method used by an AI-based commercial tool, Eunomos [3], and then we will perform a usability study in a project in financial or telecom sector in Luxembourg. To mitigate threats to the external validity, we used four different regulations and six organizational business processes in our case study. This helped illustrate that our pairwise comparison algorithm and framework is not simply specific to comparing limited regulations and business processes. Although, we provided examples from other regulations while examining the cases, in our case study the regulations we used are all regulations that exist in Ontario and that are related to healthcare. We did not analyze our framework and our comparison algorithm in other domains. To mitigate this threat, we used the Hohfeldian ontology to build the Hohfeldian models, Legal-GRL models and covered as many cases as possible for comparison. The Hohfeldian ontology is a generic ontology for analyzing regulations and identifying different

types of rights. Nevertheless, the framework needs to be validated outside the healthcare area.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we introduced a goal-oriented method for compliance with multiple regulations. We identified five cases for comparison of legal statements and provided solution on how to model them with Legal-GRL and link them to the organizational GRL models. We also established links between two Legal-GRL models to ensure the change management.

One of the constraints with goal modeling notations in general is the scalability issue, especially, when the number of elements in the models increases. We eliminate this problem by using and extending the eclipse-based tool support for URN, jUCMNav [1]. jUCMNav has the capability to create several models linked to each other through the same intentional element references. The satisfaction values for the intentional elements traverse from one model to the other via GRL analysis algorithms. These features of jUCMNav help avoiding having large, not scalable models.

In future work, we want to integrate our work with the Eunomos framework [3] [4]. Eunomos is a legal knowledge and document management system that focuses on identifying norms, cross-references and semantic similarities, with a clear structure for representing multiple interpretations and normative change. The Eunomos repository of law in legislative XML format can be integrated with the LEGAL-URN to help creating a semi-automatic method for developing Hohfeldian models. Eunomos generates a list of the most similar pieces of legislation in their database using Cosine Similarity methods. By reusing the Cosine Similarity method, we aim to identify the similar pieces of legislation at the legal statement or Hohfeldian Model level. By doing this integration, we can likely improve the comparison algorithm for multiple regulations and develop a semi-automatic algorithm.

Furthermore, we would like to improve GRL analysis algorithms to detect conflicts between the Legal-GRL models in the same way GRL performs for conflicting goals.

The comparison between regulations can also be done through the measurement of high-level goals of the regulation [28], which provide support for understanding the convergence and divergences between regulatory goals, and for resolving combined compliance problems.

ACKNOWLEDGMENT

This work was partially funded by AFR - PDR grant #5810263 and by previous scholarships from the Natural Science and Engineering Research Council of Canada.

REFERENCES

- [1] M. Adedjouma, M. Sabetzadeh and L. Briand, “Automated Detection and Resolution of Legal Cross References: Approach and A Study of Luxembourg's Legislation”, In 22nd IEEE Int. RE Conf. (RE'14), IEEE CS, 2014, to appear.
- [2] D. Amyot, G. Mussbacher, S. Ghanavati, and J. Kealey, “GRL modeling and analysis with JUCMNav”, Proc. of the 5th Int. i* Workshop (iStar'11), CEUR-WS, Vol-766, 2011, pp. 160-162.
- [3] G. Boella, M. Martin, P. Rossi, L. van der Torre, and A. Violato. “Eunomos, a legal document and knowledge management system for regulatory compliance”, In ITAIS Conf., Springer 2012, pp. 571-578.
- [4] G. Boella, S. Colombo Tosatto, S. Ghanavati, J. Hulstijn, L. Humphreys, R. Muthuri, A. Rifaut, and L. van der Torre, “Integrating Legal-URN and Eunomos: Towards a Comprehensive Compliance Management Solution”, In AI and the Complexity of Legal Systems (AICOL), 2014. To appear.
- [5] T. Breaux, A. Antón, K. Boucher and M. Dorfman, “Legal requirements, compliance and practice: An industry case study in accessibility” In 16th IEEE RE'08, IEEE CS, 2008, pp. 43-52.
- [6] Department of Justice, “Privacy act (r.s.c., 1985, c. p-21)”, <http://laws-lois.justice.gc.ca/eng/acts/P-21/index.html>, 1985.
- [7] S. Easterbrook, and B. Nuseibeh, “Managing inconsistencies in an evolving specification”, In Requirements Engineering, Proc. of the 2nd IEEE Int. Symposium, IEEE CS, 1995, pp. 48-55.
- [8] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh, “Inconsistency handling in multiperspective specifications”, IEEE Trans. Software Eng., 1994, pp. 569-578.
- [9] S. Ghanavati, D. Amyot, and L. Peyton, “Towards a framework for tracking legal compliance in healthcare”, In CAiSE07, v. 4495 of LNBP, Springer, 2007, pp. 218-232.
- [10] S. Ghanavati, D. Amyot, and L. Peyton, “Compliance analysis based on a goal-oriented requirement language evaluation methodology”, In RE09, IEEE CS, 2009, pp. 133-142.
- [11] S. Ghanavati, D. Amyot, A. Rifaut, “Legal Goal-oriented Requirement Language (Legal GRL) for Modeling Regulations”, Workshop on Modeling in Software Engineering @ICSE2014 (MiSE), ACM, 2014, pp. 1-6.
- [12] S. Ghanavati, “LEGAL-URN framework for legal compliance of business processes”, Ph.D. thesis, University of Ottawa, Canada, 2013. <http://hdl.handle.net/10393/24028>
- [13] S. Ghanavati, D. Amyot, and L. Peyton, “A systematic review of goal-oriented requirements management frameworks for business process compliance”, In 4th RELAW, 2011, pp. 25-34.
- [14] S. Ghanavati, L. Humphreys, G. Boella, L. Di Caro, L. Robaldo, and L. van der Torre, “Business Process Compliance with Multiple Regulations”, 33rd Int. Conf. on Conceptual Modeling (ER'14), Springer, 2014. To appear.
- [15] D. Gordon and T. Breaux, “Comparing requirements from multiple jurisdictions” In 4th RELAW, IEEE, 2011, pp. 43-49.
- [16] D. Gordon and T. Breaux “Reconciling multi-jurisdictional legal requirements: A case study in requirements water marking”, In 20th Int. Requirements Eng. Conf., IEEE CS, 2012, pp. 91-100.
- [17] D. Gordon, and T. Breaux, “Managing multi-jurisdictional requirements in the cloud: towards a computational legal landscape”, In 3rd ACM Workshop on Cloud Computing Security, ACM, 2011, pp. 83-94.
- [18] Government of Canada, “Personal Information Protection and Electronic Documents Act (PIPEDA)”, <http://laws-lois.justice.gc.ca/eng/acts/P-8.6/index.html>, 2011.
- [19] Government of Ontario, “Health Care Consent Act (HCCA)”, 1996. http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_96h02_e.htm, 1996. [Online; accessed January 2013]
- [20] Government of Ontario, “Personal Health Information Protection Act (PHIPA)”, 2004, http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_04p03_e.htm#BK39.
- [21] Government of Ontario, “Quality of Care Information Protection Act (QCIPA)”, 2004. http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_04q03_e.htm, 2004.
- [22] ITU-T, “Recommendation Z.151 (10/12), User Requirements Notation (URN) - Language definition”, <http://www.itu.int/rec/T-REC-Z.151/en>, 2012.
- [23] N. Kiyavitskaya, A. Krausová, and N. Zannone, “Why Eliciting and Managing Legal Requirements Is Hard”, in RELAW'08, IEEE CS, 2008, pp. 26-30.
- [24] E. de Maat, R. Winkels, and T. van Engers, “Automated detection of reference structures in law,” In 19th Legal Knowledge and Information Syst., IOS Press, 2006, pp. 41-50.
- [25] J. Maxwell, A. Antón and P. Swire, “A legal cross-references taxonomy for identifying conflicting software requirements”, In 19th Requirements Engineering Conf., IEEE, 2011, pp. 197-206.
- [26] Ministry of Health and Long-Term Care, Ontario, “Municipal freedom of information and protection of privacy act” http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_90m56_e.htm#BK48, 2007.
- [27] Ministry of Health and Long-Term Care, Ontario, “Freedom of information and protection of privacy act”, http://www.e-laws.gov.on.ca/html/statutes/english/elaws_statutes_90f31_e.htm#BK63, 2011.
- [28] A. Rifaut and S. Ghanavati, “Measurement-oriented comparison of multiple regulations with GRL”, In 5th RELAW, IEEE CS, 2012, pp. 7-16.
- [29] A. Siena, I. Jureta, S. Ingolfo, A. Susi, A. Perini and J. Mylopoulos, “Capturing variability of law with Nómös 2”, In Conceptual Modeling, vol. 7532 of LNCS, 2012, pp. 383-396.
- [30] A. Siena, A. Perini, A. Susi and J. Mylopoulos, “Towards a framework for law-compliant software requirements”, In 31st Int. Conf. Software Engineering, IEEE CS, 2009, pp. 251-254.
- [31] A. van Lamsweerde, R. Darimont, and E. Letier, “Managing conflicts in goal-driven requirements engineering”, Software Engineering, IEEE Transactions on, 24(11), 1998, pp. 908-926.
- [32] L. Wenar, “Rights”, <http://plato.stanford.edu/entries/rights/>, 2004. [Online; accessed June-2014].

Identifying and Classifying Ambiguity for Regulatory Requirements

Aaron K. Massey*, Richard L. Rutledge*, Annie I. Antón*, Peter P. Swire†

*School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA

{akmassey, rrutledge, aianton}@gatech.edu

†Scheller College of Business, Georgia Institute of Technology, Atlanta, GA, USA

{Peter.Swire}@scheller.gatech.edu

Abstract—Software engineers build software systems in increasingly regulated environments, and must therefore ensure that software requirements accurately represent obligations described in laws and regulations. Prior research has shown that graduate-level software engineering students are not able to reliably determine whether software requirements meet or exceed their legal obligations and that professional software engineers are unable to accurately classify cross-references in legal texts. However, no research has determined whether software engineers are able to identify and classify important ambiguities in laws and regulations. Ambiguities in legal texts can make the difference between requirements compliance and non-compliance. Herein, we develop a ambiguity taxonomy based on software engineering, legal, and linguistic understandings of ambiguity. We examine how 17 technologists and policy analysts in a graduate-level course use this taxonomy to identify ambiguity in a legal text. We also examine the types of ambiguities they found and whether they believe those ambiguities should prevent software engineers from implementing software that complies with the legal text. Our research suggests that ambiguity is prevalent in legal texts. In 50 minutes of examination, participants in our case study identified on average 33.47 ambiguities in 104 lines of legal text using our ambiguity taxonomy as a guideline. Our analysis suggests (a) that participants used the taxonomy as intended: as a guide and (b) that the taxonomy provides adequate coverage (97.5%) of the ambiguities found in the legal text.

I. INTRODUCTION

Most people who bother with the matter at all would admit that the English language is in a bad way, but it is generally assumed that we cannot by conscious action do anything about it.

— George Orwell

Orwell's *Politics and the English Language* details ways authors conceal their actual meaning behind vague or ambiguous language. He believed much of this was due to sloppiness, and that writers could actually do something about it, but for readers of ambiguous language, rewriting is not an option. More importantly, ambiguity sometimes accurately conveys an authors intent. Legal texts are sometimes intentionally ambiguous [1]. Requirements engineers have long recognized that natural language is often ambiguous [2]. Resolving ambiguities in source documents for requirements remains an area of active research. In particular, researchers have not focused on identifying ambiguities in legal texts that govern software systems, which is critical because ambiguities in legal texts can neither be ignored nor easily removed. Many

approaches to resolving ambiguity in software requirements rely on disambiguation or removal of the ambiguity. These may simply not be an option for software engineers addressing ambiguity in a legal text. This paper explores ambiguity in a legal text from the U.S. healthcare domain and whether software engineers can actually do something about it.

Our prior research focused on compliance with the Health Insurance Portability and Accountability Act (HIPAA)¹ [3], [4]. Non-compliance with HIPAA can result in significant fines. The U.S. Department of Health and Human Services (HHS) fined WellPoint \$1.7 million², a Massachusetts healthcare provider \$1.5 million³, and Cignet Health \$4.3 million⁴ for non-compliance with HIPAA. In 2009, Congress amended HIPAA with the HITECH Act, which was passed as a part of the American Recovery and Reinvestment Act⁵. HITECH outlines a set of objectives that incentivize Electronic Health Record (EHR) systems development by providing payments to healthcare providers using EHRs with certain “meaningful uses,” which are further detailed by the U.S. Department of Health and Human Services (HHS), the federal agency charged with regulating healthcare in the United States [5].

The first step for engineers building HITECH-regulated systems is examining the text of the regulation and extracting requirements from it. Unfortunately, extracting software requirements from regulations is extremely challenging [1], [6], [7]. Even reading and understanding these documents may be beyond the capability of professional engineers [8]. Identifying ambiguous statements and understanding why those statements are ambiguous are critical skills for requirements engineers reading legal texts. Even outside of the legal domain, too much unrecognized ambiguity is considered one of the five most important reasons for failure in requirements analysis [9]. To our knowledge, this paper is the first to examine identification and classification of ambiguities in a legal text for the purpose of software requirements analysis.

Many types of ambiguities exist, and each type must be

¹Pub. L. No. 104-191, 110 Stat. 1936 (1996)

²<http://www.hhs.gov/ocr/privacy/hipaa/enforcement/examples/wellpoint-agreement.html>

³<http://www.hhs.gov/ocr/privacy/hipaa/enforcement/examples/meei-agreement.html>

⁴<https://www.huntonprivacyblog.com/2011/02/articles/hhs-fines-cignet-health-4-3-million-for-violation-of-hipaa-privacy-rule/>

⁵Pub. L. No. 111-5, 123 Stat. 115 (2009)

disambiguated differently by requirements engineers. Herein, we define an ambiguity taxonomy consisting of six broad ambiguity types based on definitions used in requirements engineering, law, and linguistics. *Lexical* ambiguity refers to a word or phrase with multiple valid meanings. *Syntactic* ambiguity refers to a sequence of words with multiple valid grammatical interpretations regardless of context. *Semantic* ambiguity refers to a sentence with more than one interpretation in its provided context. *Vagueness* refers to a statement that admits borderline cases or relative interpretation. *Incompleteness* is a grammatically correct sentence that provides too little detail to convey a specific or needed meaning. *Referential* ambiguity refers to a grammatically correct sentence with a reference that confuses the reader based on the context provided. Each of these types of ambiguity are described in more detail in Section III-A. Some types of ambiguity require additional analysis or disambiguation before implementation can begin. In Section II, we compare and contrast the approaches that prior researchers have taken to different types of ambiguity.

Ambiguities complicate reading, understanding, and examining legal texts for software requirements. Herein, we conduct a case study to determine how prevalent ambiguity is in legal texts. Our findings suggest that ambiguity is prevalent in legal text. In 50 minutes of examination, participants in our case study identified on average 33.47 ambiguities in 104 non-blank lines of legal text⁶ using our ambiguity taxonomy as a guideline. Participants did not, however, achieve a strong level of agreement on the exact number and type of ambiguity, regardless of whether we measured agreement over all participants or just over the two groups we examined (technologists and policy analysts).

The remainder of this paper is organized as follows. Section II introduces related work and background information. Section III describes our case study methodology. Section IV details the results of our case study. In Section V, we discuss the implications of this work. Section VI presents potential threats to the validity of this work. Finally, in Section VII, we summarize our work and provide directions for future work in this area.

II. RELATED WORK

The majority of software requirements specifications are written in natural language, which is inherently ambiguous and imprecise [9]. However, software engineers do not yet have a single, comprehensive, accepted definition for ambiguity [10]. Ambiguity has been defined as a statement with more than one interpretation [11]. The IEEE Recommended Practice for Software Requirements Specifications states that a requirements specification is unambiguous only when each requirement has a single interpretation [12]. Lawyers, on the other hand, depend on ambiguity to ensure that laws and regulations are not dependent on transient standards [10]. For example, lawyers might require “reasonable” encryption practices rather than

⁶We had 16 lines of legal text in our tutorial and 121 total lines in our survey. Seventeen of the lines in our survey were blank.

specifying a particular encryption algorithm or standard that might be outdated in a few years. Linguists have created detailed ambiguity classifications. In this section, we present related work on ambiguity in requirements engineering, focusing on legal requirements, and in linguistics.

A. Ambiguity in Requirements Engineering

Common sense suggests that an unambiguous statement would have only a single, clear interpretation. But how should we classify statements that have no interpretations? Vague or incomplete statements may not have a valid interpretation. For a requirements engineer, a statement that depends heavily on domain knowledge may also, at first, appear uninterpretable. Herein, we consider vague or incomplete statements to be ambiguous because they are not unambiguous. That is, we consider them to be ambiguous because they do not have a single, clear interpretation.

Requirements engineers may tolerate requirements with multiple interpretations early in the development of a new set of software requirements [13]. In addition, some statements may be *innocuous* because only one possible interpretation would be reasonable, and these statements are unlikely to lead to misunderstandings [11], [14]. Requirements with statements having more than one reasonable interpretation are *nocuous* and likely to lead to misunderstandings if not clarified [11], [14]. Legal domain knowledge would be required to differentiate between innocuous and nocuous requirements in this study. Since we do not assume our case study participants have the necessary background, we do not consider the difference between nocuous and innocuous to be meaningful. Chantree et al. make an additional distinction between *acknowledged* ambiguities, which are known to engineers, and *unacknowledged* ambiguities, which are unknown to engineers [11]. Our case study focuses only on identifying (i.e. acknowledging) ambiguity in legal texts. We consider unacknowledged ambiguity to be outside the scope of our work.

Many software engineering approaches to ambiguity involve the development of tools or techniques for recognizing or eliminating ambiguity in software requirements. For example, Gordon and Breaux use refinements to resolve potential conflicts between regulations from multiple jurisdictions [15]. Researchers have used natural language processing to detect and resolve ambiguity in software requirements [16]–[18]. Van Bussel developed a machine learning approach to detecting ambiguity in requirements specifications [19]. Popescu et al. developed a semi-automated process for reducing ambiguity in software requirements using object-oriented modeling [20]. None of these approaches focused exclusively on identifying and classifying ambiguity in legal texts to which software systems must comply.

Antón et al. examine conflicts between policy documents and software requirements [21]. Although conflicts between policy documents, legal texts, and software requirements may not necessarily be a form of ambiguity, these conflicts inspired our work in two primary ways. First, Antón et al. state that alignment between policies and software requirements must

be flawless to avoid conflicts [21]. Even potential conflict should be addressed [21]. These assertions support the use of a broad definition of ambiguity. Second, although linguists view vagueness or generality as having a single, albeit broad, meaning [22] that is sometimes used to force readers to come to their own understanding or interpretation [23], Antón et al. explicitly state that incompleteness is a form of engineering ambiguity that must be addressed for policy compliance.

B. Ambiguity in Linguistics

Empson's book on literary criticism identifies seven types of ambiguity [23]. This book led to our ambiguity taxonomy for the purposes of evaluation or criticism. Many authors use language simply to provoke a reaction in the reader, and some authors use Empson's ambiguity types for that purpose. We chose not to map Empson's concepts of discovery, incoherence, and division [23] to our taxonomy because their primary utility is for literary criticism or interpretation.

Berry et al. identified linguistic types of ambiguities [10], which they classify according to six broad types, some of which have sub-types. For example, pragmatic ambiguity includes referential ambiguity and deictic ambiguity. Their classification is similar to other classifications of linguistic ambiguity [22]. Berry et al. also examine legal ambiguity [10]. They describe the legal principles used to interpret ambiguity when encountered rather than defining ambiguity. Consider the following legal principle:

AMBIGUUM FACTUM CONTRA VENDITOREM INTERPRETANDUM EST: An ambiguous contract is to be interpreted against the seller.

This principle does not define ambiguity, rather it provides a mechanism for resolving it in contract law. This principle supports intentionally ambiguous language in legal writing by providing an context in which it can be disambiguated. Unfortunately, requirements engineers do not have the appropriate domain knowledge to interpret this language clearly, and they cannot simply ignore it or remove it. Thus, they must learn to recognize it and seek help from a legal domain expert.

Linguists and philosophers often classify ambiguity in a finer granularity than we do herein. For example, Sennet's syntactic classification ambiguity includes the subtypes phrasal, quantifier and operator scope, and pronouns [22]. Similarly, lexical ambiguity could be classified as either homonymy or polysemy [10]. Linguists and philosophers continue to debate the nature of ambiguity and correct usage of natural language [24]. In particular, classifying types of ambiguity is itself often ambiguous [25]. Even seemingly simple grammatical corrections can quickly balloon into fundamental arguments. Attempting to define what constitutes an arbitrator for "correct" usage in English is extremely challenging [26]. A discussion of the nuance involved in interpreting or correcting language use is outside the scope of this investigation.

III. CASE STUDY METHODOLOGY

Our case study methodology is based upon the Goal/Question/Metric (GQM) model [27]–[29]. The GQM

model starts with a set of goals. Each goal is addressed by at least one question, with each measured by at least one metric. Following this paradigm focuses the case study on the research questions and minimizes extraneous test participant tasks. Our research goal formulated using the GQM template is:

Analyze **empirical observations** for the purpose of **characterizing ambiguity identification and classification** with respect to **legal texts** from the viewpoint of **students in a graduate-level Privacy course** in the context of **§ 170.302 in the HITECH Act**.

Given this research goal, we formulate the following questions:

- Q1:** Does the taxonomy provide adequate coverage of the ambiguities found in § 170.302?
- Q2:** Do participants agree on the number and types of ambiguities they identify in § 170.302?
- Q3:** Do participants agree on the number and types of intentional ambiguities they identify in § 170.302?
- Q4:** Do participants agree on whether software engineers should be able to build software that complies with each paragraph of § 170.302?
- Q5:** Does an identified ambiguity affect whether participants believe that software engineers should be able to build software that complies with each paragraph of § 170.302?

The remainder of this Section is organized as follows: We first discuss important terminology, providing definitions for each ambiguity type in our taxonomy. Subsection III-B details our participant selection criteria and all materials used for this study. We introduce measures to evaluate each of these questions in Subsection III-C.

A. Terminology

Case study participants were asked to identify ambiguity in the HITECH Act, 45 CFR Subtitle A, § 170.302. We provided participants with a taxonomy that defines six separate types of ambiguity. Table I outlines these ambiguity types. Note that they are not mutually exclusive: a single sentence from a legal text may exhibit more than one ambiguity type. Although this ambiguity taxonomy is designed to be broadly applicable, it is not guaranteed to be comprehensive. Sentences may be ambiguous in ways that do not fall into one of these six types. To introduce our ambiguity taxonomy, we employ example ambiguities identified by our study participants rather than the examples used in our study tutorial, shown in Table I.

Lexical ambiguity occurs when a word or phrase has multiple valid meanings. Consider § 170.302(d): "Enable a user to electronically record, modify, and retrieve a patient's active medication list as well as medication history for longitudinal care." A medication history for longitudinal care could mean either a complete medication history in a particular arrangement or an abbreviated medication history used only for a particular purpose. A requirements engineer must disambiguate this prior to implementation. Another example: "Melissa walked to the bank." This could mean that Melissa walked to a financial institution or she walked to the edge of a river.

TABLE I
CASE STUDY AMBIGUITY TAXONOMY

| Ambiguity Type | Definition | Example |
|----------------|---|--|
| Lexical | A word or phrase with multiple valid meanings | Melissa walked to the bank. |
| Syntactic | A sequence of words with multiple valid grammatical interpretations regardless of context | Quickly read and discuss this tutorial. |
| Semantic | A sentence with more than one interpretation in its provided context | Fred and Ethel are married. |
| Vagueness | A statement that admits borderline cases or relative interpretation | Fred is tall. |
| Incompleteness | A grammatically correct sentence that provides too little detail to convey a specific or needed meaning | Combine flour, eggs, and salt to make fresh pasta. |
| Referential | A grammatically correct sentence with a reference that confuses the reader based on the context | The boy told his father about the damage. He was very upset. |

Syntactic ambiguity occurs when a sequence of words has multiple valid grammatical parsings. Consider § 170.302(f): “Enable a user to electronically record, modify, and retrieve a patient’s vital signs...” Here, “electronically” may refer to all the verbs “record, modify, and retrieve” or only to “record.” It seems unlikely that the U.S. government wants EHR vendors to “electronically modify a patient’s vital signs.” But, electronic recording or retrieving seem like reasonable requirements. Again, a requirements engineer must disambiguate prior to implementation. Also: “Quickly read and discuss this paragraph.”

Semantic ambiguity occurs when a sentence has more than one interpretation based entirely on the surrounding context. Each word in the sentence has a distinct meaning and the sentence has a single parse tree, but the correct interpretation of the sentence requires more context. Consider § 170.302(j): “Enable a user to electronically compare two or more medication lists.” Comparing two lists is reasonably clear if a context for the comparison is provided. These lists could be compared for length, cost, drug interaction, or any number of other factors. In addition, these lists could belong to the same patient or different patients, depending on the comparison’s purpose. Other examples: “Fred and Ethel are married.” and “Fred kissed his wife, and so did Bob.” Further context is needed to determine if Fred and Ethel are married to each other or separately. Nor do we know if Fred has cause to be annoyed.

Vagueness occurs when a term or statement admits borderline cases or relative interpretation. Consider § 170.302(h)(3): “Electronically attribute, associate, or link a laboratory test result to a laboratory order or patient record.” What constitutes attributing, associating, or linking? Must these records always be displayed together or would simply having an identifier and allowing a physician to find one given the other suffice? Similarly, consider: “Fred is tall.” If Fred was a North American male and 5’2” tall, then the claim is not true. If Fred was 7’0” tall, then the claim is supported. Somewhere in between lie heights that reasonable people might disagree as to constituting “tall.”

Incompleteness occurs when a statement fails to provide enough information to have a single clear interpretation. Consider § 170.302(a)(2): “Provide certain users with the

ability to adjust notifications provided for drug-drug and drug-allergy interaction checks.” This sentence omits information that would allow requirements engineers to identify which users should have this ability or what options they would have to adjust notifications. Incompleteness must be resolved for the requirements to be implemented. Similarly, “Combine flour, eggs, and salt to make fresh pasta.” omits some necessary information such as quantity of materials and techniques to be employed.

Referential ambiguity occurs when a word or phrase in a sentence cannot be said to have a clear reference. Consider § 170.302(n): “For each meaningful use objective with a percentage-based measure, electronically record the numerator and denominator...” The meaningful use objectives that use a percentage-based measure are not referenced directly, which leaves the requirements engineer to determine which objectives must comply with this legal obligation. Other examples include pronouns and their antecedents. “The boy told his father about the damage. He was very upset.” The pronoun ‘he’ could refer to either the boy or the father. Also: “There are many reasons why lawyers lie. Some are better than others.”

We created our ambiguity taxonomy based on those ambiguity types that are relevant for regulatory compliance. It is not intended to be comprehensive with respect to all types of ambiguity. A word, phrase, sentence, or paragraph with more than one meaning may not fit in our taxonomy. For this case study, participants were instructed to classify such sentence as an **Other ambiguity**.

Because we chose a section of legal text from the HITECH Act primarily for its important implications for software development, this study was not designed to guarantee that all types of ambiguities appear in the legal text. For example, there may be no referential ambiguity in the legal text. It is also possible that a paragraph from this legal text has a single interpretation with a single, clear meaning. In our taxonomy, such statements are called **Unambiguous statements**.

Requirements engineers can use our ambiguity taxonomy as a guide when evaluating legal text for ambiguity. Each statement could be evaluated for each of the six ambiguity types in sequence from the beginning to the end of the text. If no ambiguity can be found in those six categories, then the requirements engineer could consider the statement to

be unambiguous. Each discovered ambiguity could then be examined for intent. Requirements engineers may be able to disambiguate an intentional ambiguity. For example, the legal phrase “reasonable security practices” is vague, but it could be clarified by a specific government or industry security standard. Unresolved intentional ambiguities and all unintentional ambiguities must be disambiguated by a legal expert.

B. Study Participants and Materials

We selected participants for our case study from a population of all students enrolled in a graduate-level class at the Georgia Institute of Technology, entitled Privacy Technology, Policy, and Law. This course was held during the 2014 spring semester and jointly listed by the College of Computing (CoC) and the Scheller College of Business (CoB). Eighteen students elected to participate.

Our case study materials consisted of a tutorial and a survey. We conducted the tutorial in the class session prior to the survey. During the tutorial, we briefly described the motivation for this research, explained the ambiguity taxonomy, and defined each ambiguity type using illustrative examples for each ambiguity type. After a short question and answer period about the ambiguity types, we presented a worked example of a legal text similar to what the students would be asked to analyze in the survey. The example legal text consisted of a paragraph from the HIPAA⁷. This example provided participants with an experience as similar as possible to that of the survey itself and allowed us to demonstrate each of the types of annotations that might be required of the participants during the survey. During the tutorial, we did not tell the participants which section of legal text would be covered in the survey.

We chose to conduct the tutorial and the survey in consecutive class sessions to allow participants more time to understand our ambiguity taxonomy. At the beginning of the class session during which we conducted the survey, we briefly recapped the tutorial and described two examples for each ambiguity type in our taxonomy. We provided study participants 50 minutes to complete the study. The first question asked the participant to self-identify as one of the following roles:

- 1) I am a **technologist**, and I am more interested in creating, building, or engineering software systems than I am in legal compliance or business analysis.
- 2) I am a **business analyst**, and I am more interested in creating a business based on technologies than I am in building technologies.
- 3) I am a **legal analyst**, and I am more interested in regulatory compliance than I am in building technologies or in business analytics.

We selected the HITECH Act, 45 CFR Subtitle A, § 170.302, which contains 23 paragraphs, as the legal text for this study. This section specifies the certification criteria for EHRs under Meaningful Use Stage 1. Compliance with this regulation

is a required qualification for the HITECH incentives that depend upon the use of a certified EHR. Non-compliance with this regulation would result in both regulatory penalties and loss of marketplace reputation. For each paragraph, participants identified ambiguities using a response block. The response block allowed participants to identify ambiguity type(s) identified, the line number on which it was identified, and whether the participant believed it to be intentional (i.e. an ambiguity the author meant to include) or unintentional (i.e. an ambiguity that was accidentally included). The distinction between intentional and unintentional ambiguities is one of the ways that requirements engineers can determine when they must consult a legal expert to resolve the ambiguity.

We created line numbers to simplify the annotation of ambiguities in the legal text. When a paragraph within § 170.302 contained a cross reference to another section of legal text within HITECH, we provided the referenced legal text without line numbers both to provide participants additional context to disambiguate the target legal text and to indicate that cross referenced legal texts were simply provided for context. The response block contained space for each of the six ambiguity types, a space labeled ‘Other’ for ambiguities that did not fall into one of our six types, and a space labeled ‘None’ for participants to indicate that the paragraph was unambiguous. If participants identified an ambiguity in the legal text, they wrote the respective line numbers in the appropriate space. If participants believed that the ambiguity was intentional, they also circled the line numbers after writing them. Finally, for each paragraph, participants were asked to agree or disagree with the following statement: “Software engineers should be able to build software that complies with this legal text.” We call paragraphs for which participants agree with this statement “implementable,” and we call those for which participants disagree “unimplementable.” We use responses to this question to determine whether identified ambiguities in § 170.302 affect participants’ beliefs about building compliant software.

C. Study Analysis

In addition to the standard mean and variance statistics, we employ two specialized measures for describing group participant agreement. The intraclass correlation coefficient (ICC) measures the variability of a set of responses with quantitative values across N participants [30]. Because we gave the same survey to each participant and performed calculations directly upon responses without first averaging, we employ ICC with the oneway effects model and with a single measure of interest as recommended by McGraw and Wong [30]. In cases where the responses were categorical instead of quantitative, we employ Fleiss’ kappa [31]. Both measures compute inter-rater reliability for a fixed number of participants and range from inverse-correlated (-1.0) to un-correlated (0.0) to perfectly correlation (1.0). To perform the statistical computations, we used the R Project⁸ with the Interrater Reliability (IRR) package,⁹ which supports both ICC

⁷The exact paragraph used in the tutorial was 45 CFR Subtitle A, § 164.312(a).

⁸<http://www.r-project.org/>

⁹<http://cran.r-project.org/web/packages/irr/>

and Fleiss' kappa. We analyzed the collected data to answer the questions identified above in Section III as follows:

Q1 Measures: An affirmative answer to this question requires (1) high coverage of identified ambiguities by the taxonomy and (2) minimal use of the “Other” type.

Q2 Measures: We counted the number of ambiguities each participant identified per paragraph and the number and type of each ambiguity found. Since this measure is quantitative, we measured agreement with ICC.

Q3 Measures: We employed the same statistics as with Q2 with responses restricted to intentional ambiguities. That is, we counted the number of intentional ambiguities each participant identified per paragraph and the number and type of each intentional ambiguity found. Because this measure is quantitative, we measured agreement with ICC.

Q4 Measures: We tabulated participant responses to our question of whether software engineers should be able to build compliant software for each legal paragraph. Because this data is categorical, agreement was measured with Fleiss' Kappa.

Q5 Measures: For paragraphs participants believe to be unimplementable, we calculated the percentage containing identified ambiguities.

IV. CASE STUDY RESULTS

Eighteen students volunteered for our case study. Of these eighteen participants, one provided complete responses to only five of the 23 paragraphs, and we excluded those results. We accepted responses from the remaining seventeen participants, including one participant who identified 55 ambiguities in the first eleven paragraphs and none in the remaining twelve. Some participants failed to provide a response for all parts of the response block for some questions. In each case, we removed those responses from our analysis where appropriate.

As previously mentioned, we asked participants to self-identify as either a: (1) *technologist*, (2) *business analyst*, or (3) *legal analyst*. Because only one participant self-identified as a legal analyst, we combined groups (2) and (3) to produce a new group that we refer to as *policy analysts*. This resulted in a roughly equal division of our seventeen participants with nine in the technologist group and eight in the policy analyst group.

We now discuss the results of our case study for each research question discussed in Section III.

Q1: Does the taxonomy provide adequate coverage of the ambiguities found in § 170.302?

Answer: Yes, on average, the participants identified 33.47 ambiguities for the paragraphs in § 170.302, including ambiguities from each type in the taxonomy. The least frequently identified ambiguity type is Semantic with an average of 1.59. The most frequently identified type was Vagueness with an average of 9.82. The ‘Other’ type had an average of 0.82.

Both technologists and policy analysts identified ambiguities from every type in the taxonomy. Figure 1 shows the total

ambiguities identified by participants for each paragraph in § 170.302.¹⁰ It also shows the relative totals for each ambiguity type. Note that paragraph § 170.302(a), in which participants found the most ambiguities, also includes the preamble text that appears at the beginning of § 170.302 and prior to the paragraph.

Table II shows the ambiguities (mean and standard deviation) identified in each paragraph by (1) all participants, (2) technologists, or (3) policy analysts. Intentional ambiguities were relatively rare compared to unintentional ambiguities. Some paragraphs are also appear to be less ambiguous than others. For example, in Table II, paragraphs (b), (j), (o), (r), and (w) each had less than one ambiguity on average for all participants, whereas paragraphs (a), (c), (h), (f), and (n) all had over two ambiguities on average.

The taxonomy ambiguity types overlap and the decision to select one or more types is inherently subjective. Participants used all six ambiguity types more frequently than the “Other” type. Subsection III-A provides examples of each ambiguity type as identified by the participants. Our analysis suggests (a) that participants used the taxonomy as intended: as a guide and (b) that the taxonomy provides adequate coverage (97.5%) of the ambiguities found in § 170.302.

In our prior work, we sought to compare participants' understanding of the law to a consensus expert opinion on the law [7]. This comparison worked well as an evaluation technique because rules exist for interpreting laws and regulations. Thus, a correct interpretation can be differentiated from an incorrect interpretation. Unfortunately, conducting a similar comparison to assess the “correctness” of ambiguities as identified and classified by participants in this work is not possible and would be misleading if conducted. Ambiguity is subjective [26]. No absolute authority exists to interpret ambiguity [26], so there is no way to evaluate objective correctness.

Because no correct interpretation of ambiguity exists, any comparison to a consensus expert opinion would misleadingly imply that a correct interpretation exists. In the absence of an objectively correct assessment, expertise is subjective. If a reader finds a statement ambiguous, how can an “expert” prove it is clear? Similarly, if a reader fails to interpret a statement as ambiguous, how can an “expert” prove it to be so?

Analogy to other forms of communication may help illuminate the nature of ambiguity. Comedians do not get to blame their audience for not laughing at their jokes. There is no such thing as an objectively humorous statement. Writers do not get to blame their readers for not understanding their point. Clear communication is the burden of the sender, not the recipient. Thus, in the only sense possible, when participants determine a statement was ambiguous to them, it is ambiguous.

Q2: Do participants agree on the number and types of ambiguities they identified in § 170.302?

Answer: We evaluated agreement using intraclass correlation finding only slight to fair agreement between all partici-

¹⁰One participant found a total of 55 ambiguities through paragraph § 170.302(k) and none in the remainder of the legal text.

TABLE II
AMBIGUITIES IDENTIFIED IN EACH PARAGRAPH OF THE HITECH ACT, 45 CFR SUBTITLE A, § 170.302

| Type | Intent | Paragraphs (a) - (l) (mean, std dev) | | | | | | | | | | | |
|----------|--------|--------------------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| | | (a) | (b) | (c) | (d) | (e) | (f) | (g) | (h) | (i) | (j) | (k) | (l) |
| Tech | U | 2.7, 1.8 | 0.9, 0.7 | 2.3, 0.5 | 1.1, 0.7 | 1.2, 0.8 | 2.0, 1.1 | 1.0, 0.9 | 1.6, 1.2 | 0.9, 0.9 | 0.8, 0.9 | 1.3, 1.4 | 1.2, 1.0 |
| | I | 0.6, 0.5 | 0.1, 0.3 | 0.1, 0.3 | 0.0, 0.0 | 0.0, 0.0 | 0.1, 0.3 | 0.1, 0.3 | 0.7, 0.7 | 0.1, 0.3 | 0.1, 0.3 | 0.2, 0.4 | 0.1, 0.3 |
| | C | 3.2, 1.7 | 1.0, 0.8 | 2.4, 0.5 | 1.1, 0.7 | 1.2, 0.8 | 2.1, 1.1 | 1.1, 0.9 | 2.2, 0.8 | 1.0, 0.8 | 0.9, 0.9 | 1.6, 1.3 | 1.3, 1.1 |
| Policy | U | 3.5, 3.5 | 0.9, 0.6 | 1.6, 1.5 | 1.5, 1.0 | 1.3, 0.8 | 2.6, 2.0 | 1.4, 1.9 | 2.5, 2.7 | 1.9, 1.8 | 0.9, 0.8 | 1.3, 1.6 | 0.9, 0.9 |
| | I | 0.5, 0.7 | 0.0, 0.0 | 0.0, 0.0 | 0.3, 0.4 | 0.3, 0.4 | 0.1, 0.3 | 0.0, 0.0 | 0.8, 0.8 | 0.0, 0.0 | 0.1, 0.3 | 0.1, 0.3 | 0.0, 0.0 |
| | C | 4.0, 3.3 | 0.9, 0.6 | 1.6, 1.5 | 1.8, 1.1 | 1.5, 1.0 | 2.8, 2.0 | 1.4, 1.9 | 3.3, 2.4 | 1.9, 1.8 | 1.0, 0.9 | 1.4, 1.6 | 0.9, 0.9 |
| Combined | U | 3.1, 2.8 | 0.9, 0.7 | 2.0, 1.1 | 1.3, 0.9 | 1.2, 0.8 | 2.3, 1.6 | 1.2, 1.5 | 2.0, 2.1 | 1.4, 1.5 | 0.8, 0.9 | 1.3, 1.5 | 1.1, 1.0 |
| | I | 0.5, 0.6 | 0.1, 0.2 | 0.1, 0.2 | 0.1, 0.3 | 0.1, 0.3 | 0.1, 0.3 | 0.1, 0.2 | 0.7, 0.7 | 0.1, 0.2 | 0.1, 0.3 | 0.2, 0.4 | 0.1, 0.2 |
| | C | 3.6, 2.6 | 0.9, 0.7 | 2.1, 1.2 | 1.4, 1.0 | 1.4, 0.9 | 2.4, 1.6 | 1.2, 1.5 | 2.7, 1.8 | 1.4, 1.4 | 0.9, 0.9 | 1.5, 1.5 | 1.1, 1.0 |
| | | Paragraphs (m) - (w) (mean, std dev) | | | | | | | | | | | |
| | | (m) | (n) | (o) | (p) | (q) | (r) | (s) | (t) | (u) | (v) | (w) | |
| Tech | U | 1.3, 0.8 | 2.2, 1.2 | 0.6, 0.7 | 0.7, 1.1 | 1.0, 0.9 | 0.4, 0.5 | 1.0, 0.8 | 0.2, 0.4 | 1.3, 0.8 | 1.0, 0.8 | 0.7, 0.9 | |
| | I | 0.1, 0.3 | 0.3, 0.7 | 0.3, 0.5 | 1.0, 1.1 | 0.3, 0.5 | 0.1, 0.3 | 0.4, 0.7 | 0.2, 0.4 | 0.8, 0.4 | 0.3, 0.7 | 0.0, 0.0 | |
| | C | 1.4, 0.8 | 2.6, 1.3 | 0.9, 1.0 | 1.7, 0.9 | 1.3, 0.9 | 0.6, 0.7 | 1.4, 1.2 | 0.4, 0.5 | 2.1, 0.6 | 1.3, 0.7 | 0.7, 0.9 | |
| Policy | U | 1.0, 1.2 | 1.5, 1.2 | 0.5, 0.5 | 0.6, 0.7 | 0.6, 0.7 | 0.8, 0.8 | 1.0, 1.3 | 0.8, 1.4 | 0.9, 0.9 | 0.4, 0.7 | 0.5, 0.9 | |
| | I | 0.3, 0.4 | 0.0, 0.0 | 0.4, 0.7 | 0.5, 0.7 | 0.3, 0.4 | 0.0, 0.0 | 0.4, 0.5 | 0.1, 0.3 | 0.4, 0.5 | 0.3, 0.4 | 0.0, 0.0 | |
| | C | 1.3, 1.3 | 1.5, 1.2 | 0.9, 0.9 | 1.1, 1.1 | 0.9, 0.6 | 0.8, 0.8 | 1.4, 1.2 | 0.9, 1.4 | 1.3, 1.0 | 0.6, 0.7 | 0.5, 0.9 | |
| Combined | U | 1.2, 1.0 | 1.9, 1.3 | 0.5, 0.6 | 0.6, 0.9 | 0.8, 0.9 | 0.6, 0.7 | 1.0, 1.1 | 0.5, 1.0 | 1.1, 0.9 | 0.7, 0.8 | 0.6, 0.9 | |
| | I | 0.2, 0.4 | 0.2, 0.5 | 0.4, 0.6 | 0.8, 0.9 | 0.3, 0.5 | 0.1, 0.2 | 0.4, 0.6 | 0.2, 0.4 | 0.6, 0.5 | 0.3, 0.6 | 0.0, 0.0 | |
| | C | 1.4, 1.1 | 2.1, 1.3 | 0.9, 1.0 | 1.4, 1.0 | 1.1, 0.8 | 0.6, 0.8 | 1.4, 1.2 | 0.6, 1.0 | 1.7, 0.9 | 1.0, 0.8 | 0.6, 0.9 | |

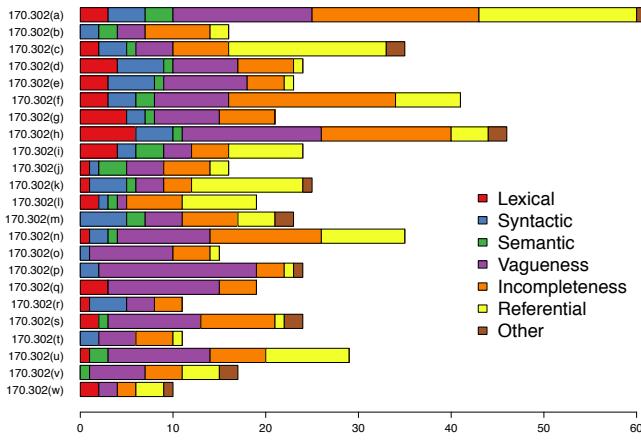


Fig. 1. Ambiguities identified in each paragraph of the HITECH Act, 45 CFR Subtitle A, § 170.302

participants on the number and types of ambiguities identified for each paragraph in § 170.302.

When examining our results according to ambiguity type, the participants demonstrate fair agreement (ICC: 0.316, $p < 0.0001$). This indicates that participants successfully identified different ambiguity types according to our taxonomy classifications. Although there is clearly room for improvement, we believe these results are encouraging given the training, time, and conditions we were able to provide our participants. If we examine participant agreement regarding whether or not each paragraph was unambiguous, we find that participants demonstrate slight agreement (FK: 0.0446, $p = 0.00288$). Participants unanimously agreed that paragraph § 170.302(h) was ambiguous and every participant except one rated § 170.302(a) as ambiguous. For the other 21 paragraphs, participants exhibited little agreement.

Figure 2 shows which ambiguity types participants identified

most. Each type (labeled on the x-axis) has two bars. The bar on the left (with hash marks) represents the number of ambiguities identified by technologists, and the one on the right (without hash marks) represents the number identified by policy analysts. Each bar is divided into two parts. The lower part (with a lighter shade) represents the proportion of the total that are unintentional ambiguities identified, and the upper part (with a darker shade) represents intentional ambiguities. For example, both technologists and policy makers identified roughly the same number of Syntactic ambiguities. In contrast, technologists and policy analysts differ in their identification of Incompleteness. Technologists identified over 100 Incompletesses, with about a quarter of those being intentional, whereas policy analysts only identified about 50 Incompletesses, most of which were unintentional.

The largest disagreement between technologists and policy analysts occurred in the Lexical and Incompleteness ambiguity types. Policy analysts found on average 4.4 times more lexical ambiguity than technologists, and technologists found 1.8 times more incompletesses than policy analysts. This may be indicative of their respective professional training and background. Lexical ambiguities are more commonly associated with grammar, writing, and linguistics, whereas Incompleteness comes primarily from software engineering. Table III details additional examples of both agreement and disagreement. Note that the number of Vaguenesses identified differs greatly, which may also be a result of training because Vagueness and Incompleteness are similar, overlapping ambiguity types. Technologists are trained to identify Incompleteness and Vagueness in formal specifications, which may carry over into identifying those ambiguity types in § 170.302.

Q3: Do participants agree on the number and types of intentional ambiguities they identified in § 170.302?

Answer: We evaluated agreement using intraclass correlation and found a slight level of agreement between participants

TABLE III
AMBIGUITIES IDENTIFIED BY TYPE

| Type | Intent | Lexical | Syntactic | Semantic | Vagueness | Incompleteness | Referential | Other |
|----------|--------|----------|-----------|----------|-----------|----------------|-------------|----------|
| Tech | U | 1.0, 1.9 | 2.9, 3.5 | 1.4, 2.2 | 5.6, 3.1 | 8.2, 5.5 | 7.3, 4.6 | 1.0, 1.2 |
| | I | 0.0, 0.0 | 0.0, 0.0 | 0.0, 0.0 | 3.0, 2.8 | 3.2, 5.7 | 0.0, 0.0 | 0.0, 0.0 |
| | C | 1.0, 1.9 | 2.9, 3.5 | 1.4, 2.2 | 8.6, 3.1 | 11.4, 10.3 | 7.3, 4.6 | 1.0, 1.2 |
| Policy | U | 3.9, 5.0 | 3.3, 3.3 | 1.6, 3.0 | 7.8, 8.0 | 6.1, 2.7 | 5.4, 4.7 | 0.6, 1.0 |
| | I | 0.5, 1.0 | 0.0, 0.0 | 0.1, 0.3 | 3.5, 4.2 | 0.1, 0.3 | 0.4, 1.0 | 0.0, 0.0 |
| | C | 4.4, 5.0 | 3.3, 3.3 | 1.8, 2.9 | 11.3, 9.0 | 6.3, 2.7 | 5.8, 4.4 | 0.6, 1.0 |
| Combined | U | 2.4, 4.0 | 3.1, 3.4 | 1.5, 2.6 | 6.6, 6.0 | 7.2, 4.5 | 6.4, 4.8 | 0.8, 1.1 |
| | I | 0.2, 0.7 | 0.0, 0.0 | 0.1, 0.2 | 3.2, 3.5 | 1.8, 4.5 | 0.2, 0.7 | 0.0, 0.0 |
| | C | 2.6, 4.1 | 3.1, 3.4 | 1.6, 2.6 | 9.8, 6.7 | 9.0, 8.2 | 6.6, 4.6 | 0.8, 1.1 |

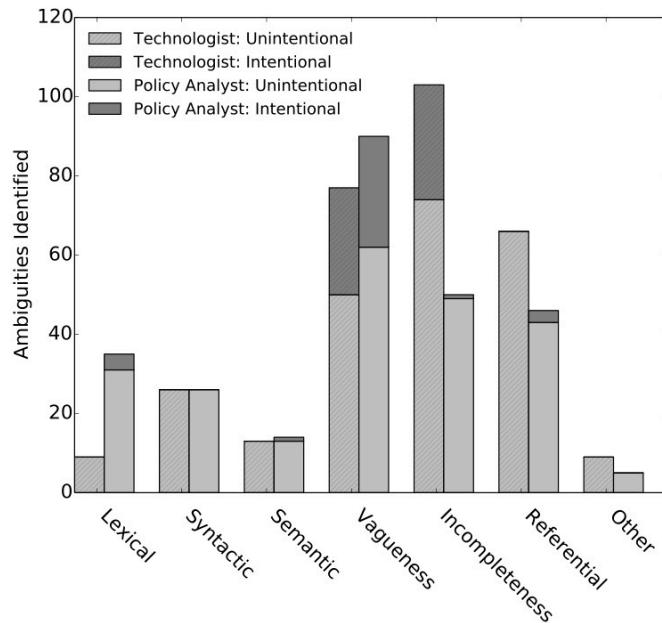


Fig. 2. Ambiguities Identified by Type

on the number and types of intentional ambiguities identified for each paragraph in § 170.302.

Participants agreed less on the number and type of intentional ambiguities than they did on the number and type of total ambiguities. Participants exhibited slight agreement on intentional ambiguities, whether measured by number (ICC: 0.141, $p < 0.0001$) or type (ICC: 0.201, $p < 0.0001$). The set of intentional Incompletences identified by the participants drove this difference. Table III shows that technologists identified an average of 3.2 intentional Incompletences compared to a 0.1 average for policy analysts. If we remove all incompletences from the calculation, the level of agreement for the number of identified ambiguities is roughly the same as before (ICC 0.134, $p < 0.0001$) and the level of agreement for the ambiguity type increases (ICC: 0.39, $p < 0.0001$).

Regardless of the agreement level, the fact that participants of both groups were able to identify intentional ambiguities at all is important because intentional ambiguity is a fundamental part of legal texts [1]. Intentional ambiguity holds important implications for software requirements that must comply with

laws and regulations [1], [3], [6], [7], [10], [32]. Consider § 170.302(p), which reads as follows:

(p) Emergency access. Permit authorized users (who are authorized for emergency situations) to access electronic health information during an emergency.

This paragraph describes the “break the glass” scenario in which physicians otherwise unable to access certain health records would be allowed access. The definition of an “emergency situation” or what it means to be “authorized for emergency situations” is not provided. Thirteen participants flagged this as an intentional ambiguity. This recognition is important because intentional ambiguities must first be identified before they may be disambiguated. Moreover, they must be periodically reevaluated regarding ambiguity and the author’s intent.

Q4: Do participants agree on whether software engineers should be able to build software that complies with each paragraph in § 170.302?

Answer: We evaluated agreement using Fleiss’ kappa and did not find agreement between participants on whether paragraphs from § 170.302 were implementable.

Participant agreement was not statistically significant for the group as a whole (FK: 0.0052, $p = 0.788$) or for the technologists as a group (0.0455, $p = 0.116$). The policy analysts disagreed slightly on the legal text’s implementability (FK: -0.124, $p = 0.0111$). This is consistent with other findings for similar tasks involving the evaluation of legal texts for software engineering purposes. Prior research notes that determining whether software requirements have met or exceeded their legal obligations is challenging [7]. Maxwell found identification and classification of legal cross references to be similarly challenging for professional engineers [6].

Q5: Does an identified ambiguity affect whether participants believe that software engineers should be able to build software that complies with each paragraph in § 170.302?

Answer: Yes, 89% of unimplementable paragraphs contained an unintended ambiguity, whereas only 48% of implementable paragraphs contained an ambiguity.

Of the 83 paragraphs found to be unimplementable by the participants, 74 contained unintentional ambiguities. Of the 216 paragraphs found to be implementable, 104 contained unintentional ambiguities. We expected that those paragraphs that Participants identified as implementable would only rarely

contain unintentional ambiguities, but our results indicate that 48% of the implementable paragraphs were deemed to contain unintended ambiguities. Prior research has shown for a similar task (identifying legally implementation-ready requirements) that individuals working alone tend to be too liberal (i.e. accept as implementation-ready requirements that need further refinement) and groups working together tend to be too conservative (i.e. reject requirements that have actually met their legal obligations) [7], [33].

V. DISCUSSION

Perhaps the most interesting results for this case study are the qualitative results. When preparing the materials for this case study, we examined § 170.302 many times to identify and classify its ambiguities. Our participants were given 50 minutes to accomplish the same task, yet they found several subtle ambiguities that eluded us. For example, consider § 170.302(q):

(q) Automatic log-off. Terminate an electronic session after a predetermined time of inactivity.

We found the phrase “predetermined time of inactivity” to be incomplete or perhaps vague because no purpose is stated; making it a challenge to determine how much inactivity is allowable. It could be considered vague because “after” admits borderline or relative cases: does it mean immediately after or at some point after? Most participants identified the statement in the same way, but one participant identified “time” as lexically ambiguous. It could mean either duration or a time of day. If interpreted as the latter, then it could be interpreted as requiring EHRs to terminate an electronic session after closing time.

Paragraph § 170.302(o) provides another interesting example:

(o) Access control. Assign a unique name and/or number for identifying and tracking user identity and establish controls that permit only authorized users to access electronic health information.

Two respondents found this paragraph to be both unambiguous and also not implementable. How could an unambiguous statement be unimplementable? This may seem unintuitive at first, but the halting problem can be stated unambiguously and cannot be implemented. Similarly, the absolute nature of the phrase “permit only authorized users to access electronic health information” could be interpreted as impossible to implement because it is not a wholly technological problem.

VI. THREATS TO VALIDITY

Case study research is incomplete without a discussion of concerns that may threaten results validity. *Internal validity* refers to the causal inferences made based on experimental data [34]. Herein, we do not attempt to determine causality for any part of this research. Our goal is simply to determine whether and how participants identify and classify ambiguity in legal texts.

Construct validity refers to the appropriate use of evaluation metrics and measures [34]. We specifically avoided the use of absolute measures of ambiguity to conform with the term as expressed in accepted IEEE standards [12]. To calculate other statistical measures, we used accepted statistics for

agreement (ICC and Fleiss’ kappa) and scrupulously followed recommended practices in applying them. Our case study participants may have become fatigued and stopped responding to the questions in our survey. To mitigate the impact of survey fatigue, we adjusted our statistical measures to account for the three surveys that contained unanswered questions.

Providing participants with only a single section of the HITECH Act and the text of cross-references contained within it is another threat to construct validity. The complete text would have unreasonably increased participant fatigue. Note that providing additional text could allow participants to either disambiguate ambiguities identified in our study or discover additional ambiguities resulting from potentially conflicting material.

External validity refers to the ability to generalize the findings to other domains [34]. We have mitigated threats to external validity by selecting a section in the HITECH Act that is representative of the style, tone, and wording of obligations found in the rest of the act. In addition, we chose a participant population with as many different backgrounds as possible rather than limiting our research to stakeholders with an engineering background. Unfortunately, two important threats of this type remain. First, our study employs a small population of students rather than a large population of practitioners. Although the findings of our study align with similar case studies that examine legal texts for engineering purposes [6], [7], students enrolled in a graduate class may not be representative of practicing engineers, lawyers, managers, and policy makers. Second, we selected a legal text from a single domain. Healthcare is a popular domain for regulatory compliance software engineering research, but other domains, like finance, also have extensive regulatory requirements. To address these threats, we plan to adapt what we have learned from this study for a broader, web-based examination of ambiguity identification and classification for multiple legal domains in the future.

Reliability refers to the ability of other researchers to replicate this methodology. We assiduously detailed both our methods and our evaluation techniques. In addition, we have made our case study tutorial and survey materials available online for researchers interested in replicating our results.¹¹ We do not believe reliability is a serious concern for this research.

VII. SUMMARY AND FUTURE WORK

The development of methods to improve and demonstrate legal compliance with federal privacy and security regulations in software systems is critical. Stakeholders of regulated software systems, and in particular requirements engineers, must be able to identify ambiguities in legal text and understand their implications for software systems. To this end, we created a taxonomy with six ambiguity types intended to encompass a broad definition of ambiguity within the context of legal texts. We conducted a case study to examine how students

¹¹<http://www.cc.gatech.edu/~akmassey/documents/ambiguity-case-study-materials.pdf>

in a graduate privacy class identify and classify ambiguity for § 170.302 in the HITECH Act. Our research suggests that ambiguity is prevalent in legal texts. In 50 minutes of examination, participants in our case study identified on average 33.47 ambiguities in 104 lines of legal text using our ambiguity taxonomy as a guide.

Participants did not exhibit strong agreement on the number and type of ambiguities present in the legal text. This may be due to the 50-minute time limit or to the complexity of the task. Our analysis suggests (a) that participants used the taxonomy as intended: as a guide and (b) that the taxonomy provides adequate coverage (97.5%) of the ambiguities found in §170.302. This suggests that the ambiguity taxonomy is sufficient for analyzing this particular legal text.

Participants were willing to accept paragraphs with unintentional ambiguities as implementable (i.e. as something for which software engineers should be able to build compliant software). Prior research has shown that software engineers are ill-equipped to perform similar tasks [6], [7]. Further research is needed in this area to provide better guidance and improve decision-making in this area.

We plan to conduct additional case studies on larger populations to better understand ambiguity in legal texts and its implications for software engineering. In particular, we seek to conduct a larger online case study covering healthcare, finance, and other regulated domains. A larger study would allow us to evaluate multiple possible aids for identifying and classifying ambiguity in legal text. In addition, we plan to examine whether identifying and classifying ambiguity improves software engineering assessments of legal implementation readiness, which our prior work has shown to be extremely challenging for engineers to do with accuracy [7].

REFERENCES

- [1] P. N. Otto and A. I. Antón, "Addressing Legal Requirements in Requirements Engineering," *15th IEEE International Requirements Engineering Conference*, pp. 5–14, 15-19 Oct. 2007.
- [2] G. Roman, "A taxonomy of current issues in requirements engineering," *Computer*, vol. 18, no. 4, pp. 14–23, Apr. 1985.
- [3] A. K. Massey, P. N. Otto, L. J. Hayward, and A. I. Antón, "Evaluating Existing Security and Privacy Requirements for Legal Compliance," *Requirements Engineering*, 2010.
- [4] A. K. Massey, P. N. Otto, and A. I. Antón, "Legal Requirements Prioritization," *Proc. of the 2nd Intl. IEEE Workshop on Requirements Engineering and the Law*, 2009.
- [5] Department of Health and Human Services, "Medicare and Medicaid Programs; Electronic Health Record Incentive Program; Final Rule," *Federal Register*, vol. 75, no. 8, July 28 2010.
- [6] J. C. Maxwell, "Reasoning about legal text evolution for regulatory compliance in software systems," Ph.D. dissertation, North Carolina State University, 2013.
- [7] A. Massey, B. Smith, P. Otto, and A. Anton, "Assessing the Accuracy of Legal Implementation Readiness Decisions," in *19th IEEE International Requirements Engineering Conference (RE)*, September 2011, pp. 207–216.
- [8] A. K. Massey, J. Eisenstein, A. I. Antón, and P. Swire, "Automated Text Mining for Requirements Analysis of Policy Documents," *21st International Conference on Requirements Engineering*, 2013.
- [9] D. M. Berry and E. Kamsties, "Ambiguity in requirements specification," in *Perspectives on Software Requirements*, ser. The Springer International Series in Engineering and Computer Science, P. Leite, J. C. Sampaio, and J. H. Doorn, Eds. Springer US, 2004, vol. 753, pp. 7–44.
- [10] D. M. Berry, E. Kamsties, and M. M. Krieger, "From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity," School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, Tech. Rep., November 2003.
- [11] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *Requirements Engineering, 14th IEEE International Conference*, 2006, pp. 59–68.
- [12] IEEE, *ANSI/IEEE Standard 830-1993: Recommended Practice for Software Requirements Specifications*. Institute of Electrical and Electronics Engineering, New York, NY, 1993.
- [13] S. Easterbrook and B. Nuseibeh, "Managing inconsistencies in an evolving specification," *Second International Symposium on Requirements Engineering*, 1995.
- [14] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Extending nocuous ambiguity analysis for anaphora in natural language requirements," in *18th IEEE International Requirements Engineering Conference*, 2010, pp. 25–34.
- [15] D. Gordon and T. Breaux, "A cross-domain empirical study and legal evaluation of the requirements watermark method," *Requirements Engineering*, vol. 18, no. 2, pp. 147–173, 2013.
- [16] M. Osborne and C. MacNish, "Processing natural language software requirement specifications," in *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, 1996, pp. 229–236.
- [17] A. Umber and I. Bajwa, "Minimizing ambiguity in natural language software requirements specification," in *Sixth International Conference on Digital Information Management*, 2011, pp. 102–107.
- [18] A. Nigam, N. Arya, B. Nigam, and D. Jain, "Tool for Automatic Discovery of Ambiguity in Requirements," *International Journal of Computer Science*, vol. 9, no. 5, September 2012.
- [19] D. van Bussel, "Detecting ambiguity in requirements specifications," Master's thesis, Tilburg University, August 2009.
- [20] D. Popescu, S. Rugaber, N. Medvidovic, and D. Berry, "Reducing ambiguities in requirements specifications via automatically created object-oriented models," in *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs*, ser. Lecture Notes in Computer Science, B. Paech and C. Martell, Eds. Springer Berlin Heidelberg, 2008, vol. 5320, pp. 103–124.
- [21] A. I. Antón, J. B. Earp, and R. A. Carter, "Precluding incongruous behavior by aligning software requirements with security and privacy policies," *Information and Software Technology*, vol. 45, no. 14, pp. 967–977, 2003.
- [22] A. Sennet, "Ambiguity," in *The Stanford Encyclopedia of Philosophy*, summer 2011 ed., E. N. Zalta, Ed., 2011.
- [23] W. Empson, *Seven Types of Ambiguity*. New Directions, 1966.
- [24] T. Wasow, A. Perfors, and D. Beaver, *Morphology and The Web of Grammar: Essays in Memory of Steven G. Lapointe*. CSLI Publications, 2005, ch. The Puzzle of Ambiguity.
- [25] T. Wasow, "Ambiguity avoidance is overrated," *to appear in a volume edited by Susanne Winkler, http://www.stanford.edu/~wasow/Ambiguity.pdf*, 2014.
- [26] D. F. Wallace, "Tense present," *Harper's Magazine*, April 2001.
- [27] V. Basili and H. Rombach, "The tame project: towards improvement-oriented software environments," *IEEE Transactions on Software Engineering*, vol. 14, pp. 758–773, 1988.
- [28] V. Basili, G. Caldiera, and D. Rombach, "The Goal Question Metric Approach," *Encyclopedia of Software Engineering*, vol. 2, pp. 528–532, 1994.
- [29] R. V. Solingen and E. Berghout, *The Goal/Question/Metric Method*. McGraw-Hill Education, 1999.
- [30] K. O. McGraw and S. P. Wong, "Forming inferences about some intraclass correlation coefficients," *Psychological Methods*, vol. 1, pp. 30–46, 1996.
- [31] J. L. Fleiss and J. Cohen, "The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability," *Educational and Psychological Measurement*, vol. 33, pp. 613–619, 1973.
- [32] J. C. Maxwell and A. I. Antón, "Discovering Conflicting Software Requirements by Analyzing Legal Cross-References," *(In Submission) IEEE International Requirements Engineering Conference*, 2010.
- [33] A. K. Massey, "Legal requirements metrics for compliance analysis," Ph.D. dissertation, North Carolina State University, 2012.
- [34] R. K. Yin, *Case Study Research: Design and Methods*, 3rd ed., ser. Applied Social Research Methods Series, L. Bickman and D. J. Rog, Eds. Sage Publications, 2003, vol. 5.

An Approach for Decision Support on the Uncertainty in Feature Model Evolution

Le Minh Sang Tran and Fabio Massacci

DISI, University of Trento

Povo, Trento, Italy

{leminhsang.tran, fabio.massacci}@unitn.it

Abstract—Software systems could be seen as a hierarchy of features which are evolving due to the dynamic of the working environments. The companies who build software thus need to make an appropriate strategy, which takes into consideration of such dynamic, to select features to be implemented. In this work, we propose an approach to facilitate such selection by providing a means to capture the uncertainty of evolution in feature models. We also provide two analyses to support the decision makers. The approach is exemplified in the Smart Grid scenario.

Index Terms—evolution, product lines, feature model evolution, variability, survivability, cost of reparation.

I. INTRODUCTION

Feature-oriented modeling [17], firstly coined by Kang *et al.* in 1990, is a method capturing a software system as a hierarchy of features. It has been widely adopted in many scenarios of software product lines engineering [2] by the term *feature models*. Feature models provide a compact representation of different variants of software systems. This enables customers' personalization by deciding which features to be included or discarded in the final product.

Software keeps evolving due to various reasons (*e.g.*, compliance with new laws or standards, or changes in market needs) and companies who ship software need a long-term strategy for their products while still delivering products with the features required at present time. It is therefore necessary to have an appropriate reasoning on the evolution of software features to support the long-term planning strategy.

A key observation underpinning our work is that apart from unpredictable black swans, many of changes could be anticipated with some degree of belief because they are the results of long-term processes. Such predictions could be made with the support of expert knowledge in a particular domain of application. A paradigmatic example in the domain of power supply is the development of a Smart Grid which is an infrastructure to efficiently manage the transmission and distribution of electricity. Different expected features and development roadmap of Smart Grid have been identified [15], [20] and the important question is how to build Smart Grid software with the right features from the beginning such that it is more resilient to evolution, and the cost of adding new features in future would be minimal. There are not many studies [7], [23], [33], [28], [12], [14] concerning the evolution of software product lines represented by feature models, although there

is a common consensus that this evolution is an important aspect [26], [6]. In particular, no existing work takes into consideration the uncertainty of feature model evolution.

In [30], we introduced a generic approach that captured requirements evolution by evolution rules, and two metrics to support the selection of a design alternative. In this work, we extend that approach to support modeling and reasoning on the uncertainty of feature model evolution. Our approach assists the selection of an *optimal configuration*, which is a set of features to be implemented. The system built on this optimal configuration could be able to survive as long as possible, and requires less effort to repair if evolution occurs. Our novel contributions from [30] include:

- a conceptual model for modeling evolution on feature models, which consists of two kinds of models (*i*) Evolution Possibility Model (ePM) describes potential possibilities a feature model could evolve, and (*ii*) Evolutionary Feature Model (eFM) describes the feature model with all changes due to evolution incorporated. Evolution is studied within a study period T which is divided into several milestones t_i . At every milestone, different possibilities of the original feature model are analyzed.
- two analysis techniques facilitate the decision support: *Survivability analysis* answers whether a configuration (*i.e.*, set of features) could survive during the expected evolution. *Repair cost analysis*: it is likely that few configurations remain valid for the entire study period due to changes, we need to repair them to make them valid. The question is which configuration requires less effort to get repaired. The former analysis takes the metrics' idea in [30], specializes it for feature models, and makes it a function of time to capture the survivability of a configuration over time.
- a self-validation of the proposed approach on the Smart Grid scenario, taken from the NESSoS European project.

This work aims to deal with anticipated evolution in feature models. For the uncertainty and evolution in requirements, readers could find a more detailed discussion in [8], [25], [9].

In the next, Section II presents the feature model background and related studies on feature model evolution. We discuss different evolution perspectives in Section III. Our approach is detailed in Section IV and Section V. Section VI

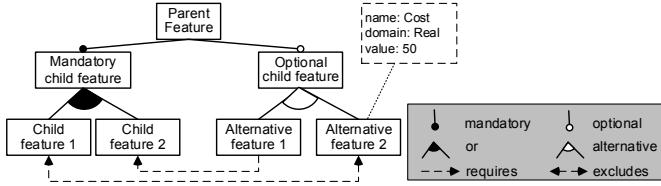


Fig. 1. An example of a feature model.

applies the proposed approach in a Smart Grid scenario. Finally, we provide an extended discussion about the approach in Section VII and conclude the paper in Section VIII.

II. BACKGROUND

Previous work related to this study could fall into two categories: studies about analyses on feature models, and studies about the evolution of feature models. For the first category, readers are referred to [2] for a systematic survey. Here, we only present a background about feature models, and studies in the latter category.

A. Feature Model

Feature model has many variants, but its basic form includes hierarchical features and relations between them [2]. The top feature represents the software system which is implemented by its children. A *configuration* is the set of features that implement the top feature. The basic feature model could be extended to include more information about the features via *attributes*. An *attribute* typically should consist of a name, a domain, and a value. The basic relations are:

- *Mandatory*: the selection of a parent feature requires the selection of its *mandatory* child features.
- *Optional*: the selection of a parent feature optionally leads to the selection of its *optional* child features.
- *Alternative (xor)*: when a parent feature is selected, exactly one of its *alternative* children ought to be selected.
- *Or*: when the parent feature is selected, at least one or more of its *or* children must be selected.
- *Requires*: a feature A requires a feature B if the selection of A implies the selection of B.
- *Excludes*: a feature A excludes a feature B if they are cannot be part of a same configuration.

Extended feature models, Fig. 1, can include complex constraints like “if an attribute of feature A is lower than a value, then feature B cannot be part of the configuration” [2].

B. Related Work on Feature Model Evolution

Botterweck et al. [7] report the need of companies to have a long-term strategy and plan product portfolios in the years ahead. They apply feature models to plan changes; evolution is considered as a sequence of feature models. They propose the Evolution Plan and Evolution Feature Model (EvoFM). The former provides an overview of evolution steps across time. The latter extends an ordinary feature model with some sub-features to support the consideration of evolution options; each EvoFM instance expresses an evolution step. Later, Pleuss et

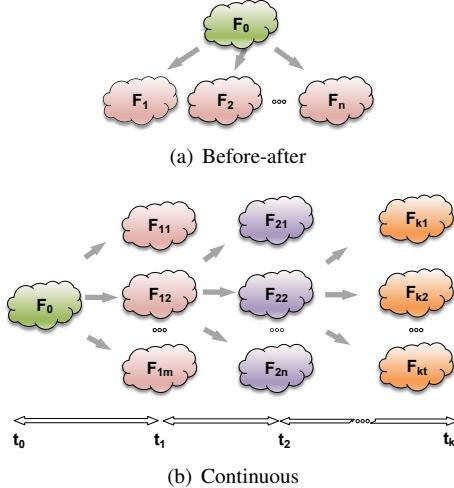


Fig. 2. Two kinds of evolution perspective of feature models.

al. [23] envelop these artifacts into the EvoPL framework to plan and manage the evolution of feature models.

Other studies about the impact of evolution in feature models are [33], [28], [12], [14]. In these studies, evolution is captured by the addition, removal, modification of entities in feature models. Thüm et al. [28] study the impact of changes by presenting an algorithm to classify changes (e.g., refactoring, specialization, generalization). Gamez and Fluentes [12] work on cardinality-based feature models and evaluate the change impact by computing the differences between previous configuration and new evolved configuration. Ye and Zhang [33] provide formal specifications to detect any conflicting dependencies introduced by adding or removing features. Guo et al. [14] also analyze the impact of evolution on the consistency of feature models. They provide an ontology-based formalization of feature models and a set of consistency constraints to identify inconsistencies in feature model evolution.

In our work, evolution also happens in time as a sequence of feature models. We also capture the evolution uncertainty and provide reasoning techniques to select an optimal configuration while they do not do it.

Besides, there are a number of studies that have addressed the issues of uncertainty modeling of requirements and architecture. Our own work [30] addresses the problem of the known unknowns. Paper [9] tackles the issue of adaptation when the domain is known but the events are unknown. See also [8], [25] for additional references.

III. EVOLUTION PERSPECTIVES

Evolution appears on feature models due to changes from the environment. Changes possibly compete with each other, for example, to do with raising standards. This is also mentioned as *multiple change*, one of the key issues in evolution, identified by Lam and Loomes [19]. The likely winners cannot be identified precisely, but could be foreseen at some levels of certainty based on expert knowledge in the system domain. This uncertainty should be addressed in order to improve the sustainability of the system.

Evolution is usually examined in a determined and finite study period. Evolution can be analyzed at once during this period *i.e.*, how the feature model looks like at the beginning – the *before* model, and how it potentially looks like at the end – the *after* ones. Many *after* models with different likelihood of occurrence are allowed to capture the evolution uncertainty. However, exactly one *after* model will occur in reality. We call this evolution perspective *before-after evolution*.

A different perspective divides the study period into several milestones at which we study how the feature model looks like. At the beginning (milestone t_0) the feature model appears with no change – the *original* feature model. At milestone t_1 , the original model could evolve into some *after₁* models with some probability, similar to the above *before-after* evolution. These models could further evolve into some other *after₂* models at milestone t_2 and so on. We call this kind of evolution perspective *continuous evolution*.

Fig. 2 visualizes these two kinds of evolution perspective where a feature model is depicted as a cloud. Fig. 2(a) illustrates the *before-after* evolution perspective where a feature model might evolve into other models. Fig. 2(b) exemplifies the *continuous evolution* perspective.

Clearly, the *before-after evolution* perspective is a special case of the *continuous evolution* perspective where there is only a single milestone in the entire study period. Moreover, the long-term usage period of a system in practice is often over years. During such period, many changes might occur, and the winner might depend on the previous ones. The *continuous evolution* perspective thus looks better than its sibling to analyze the evolution in long-term period. Thus we aim to support continuous evolution of feature models.

IV. MODELING THE FEATURE MODEL EVOLUTION

This section discusses our proposed approach to model the evolution in feature models. The EVE framework [19] categorized changes related to evolution into four types: environmental changes (*e.g.*, the introduction of new laws), requirement changes (*e.g.*, new requirements derived from environmental changes), viewpoint changes (*e.g.*, introduction of a new technology), and design changes (*e.g.*, introduction/removal of a feature). Among those, design changes reflect changes in an original feature model. The first three types of changes are mostly the original cause behind design changes.

Evolution in feature models is captured by two kinds of models: *Evolution Possibility Model* (*ePM*) and *Evolutionary Feature Model* (*eFM*). The *ePM* captures changes outside a feature model, a.k.a external changes, *i.e.*, environmental changes, requirements changes, and viewpoint changes. These changes will incite design changes in a feature model, which are captured by the *eFM*.

A. Evolution Possibility Model

The Evolution Possibility Model (*ePM*) captures all possible anticipated external changes during a study period, and arranges them into potential situations that might occur in future due to evolution. The *ePM* model enables the traceability

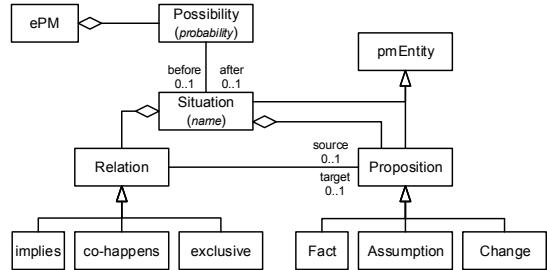


Fig. 3. The meta-model of Evolution Possibility Model.

of external changes and design changes. It helps to answer questions like “*why should this new feature be introduced?*”.

Fig. 3 presents the meta-model for an *ePM*. The *ePM* consists of a set of *Possibilities*. A *Possibility* captures how (external) changes happen. It describes the *Situations* that exist before, and after changes happen. The likelihood that the corresponding changes happen is depicted as the *probability* attribute of the *Possibility*. A *pmEntity* represents an entity (either *Situation* or *Proposition*) in an *ePM* model. It is used to link between an *ePM* model and an *eFM* model. A *Situation* consists of a set of *Propositions* which are either *Facts*, or *Assumptions*, or potential *Changes*. Each *Situation* has a *name* to distinguish itself among others. The *Relation* between these *Propositions* includes: 1) *implies* relation denotes a likely causal relation between two *Propositions*. For example, a *Fact*, or an *Assumption* might incite a *Change*; or a *Change* could also incite another *Change*. 2) *co-happens* relation denotes that two *Changes* should happen together. 3) *exclusive* relation denotes the mutual exclusion between two *Changes*. If one change happens, the other ones will not.

The situation where no change exists is referred to as *before* situation, and the others are referred to as *after* situations. For every situation, we could always construct a corresponding feature model. The feature models of *before*, and *after* situations are respectively referred to as *before*, and *after* feature model. Hereafter we abuse the name of a situation for its corresponding feature model, and versa.

In order to formally express the evolution possibilities, we adopt the *observable rule* proposed in [30] for dealing with requirements evolution. It has been adopted to address the evolution of risks in long-lived software system [29].

Let F be a situation, and F_i be *after_i* situations which F might evolve into. The observable rule r_o of F is below:

$$r_o(F) = \left\{ F \xrightarrow{p_i} F_i \mid \sum_{i=1}^n p_i = 1 \right\} \quad (1)$$

where p_i is the *evolution probability* that F evolves into F_i . We assume all known possibilities are anticipated and mutually exclusive. Additionally, we ignore all unknown possibilities. Therefore the sum of all p_i equals 1.

Loosely speaking, *ePM* could be seen as a collection of observable rules where each situation has at most one observable rule. In an observable rule, we call the situation

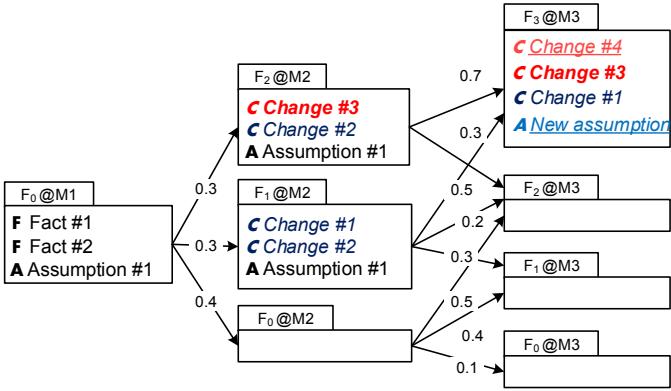


Fig. 4. An example of Evolution Possibility Model

on the left hand side of the arrow as *before* situation, and the ones on the right hand side as *after* situations. A situation could be an *after* situation in an observable rule, and could be a *before* situation in another observable rule. The situation which is not an *after* situation in any observable rules is also referred to as the *original* situation. Similarly, the situations which are not *before* situations in any rules are referred to as *leaf* situations. We additionally use the empty set notation (\emptyset) to denote the observable rules of *leaf* situations as follows:

$$r_o(F) = \emptyset \Leftrightarrow F \text{ is a leaf situation} \quad (2)$$

Example 1. Fig. 4 illustrates an example of *ePM*. In this figure, situations are illustrated as folder shapes where situation names are on the top, followed by the milestones, the situation's propositions are located inside. An evolution possibility is depicted by an arrow connecting a *before* situation to an *after* situation. Evolution probability is a label decorated to the arrow. To keep the *ePM* simple, this example does not consider relations among propositions, but presents them as a list. In this list, the first letter denotes the proposition type, *i.e.*, F – fact, A – assumption, C – change. The original situation is $F_0 @ M1$, and the leaf situations are $F_i @ M3$ ■

B. Evolutionary Feature Model

The Evolutionary Feature Model (*eFM*) captures all feature models in all situations of the *ePM* model. The *eFM* incorporates all design changes due to evolution. Syntactically, the *eFM* is an attributed feature model which is enriched with a new relation, namely *traces*, connecting an entity in *ePM* (*i.e.*, a situation, or a proposition) to an entity (*i.e.*, a feature, or a feature attribute, or a complex constraint) in *eFM*. By modeling this way, design changes made to a feature model are directly or indirectly associated with situations in *ePM*.

Fig. 5 presents the meta-model of the *eFM*. The meta-model includes notions for a basic attributed feature model, and a new type of relation – *traces*. An *eFM* consists of a set of *Elements*. An *Element* could be either an *fmEntity*, or an *fmRelation*, or a *traces* relation. An *fmEntity* represents a *Feature*, or an

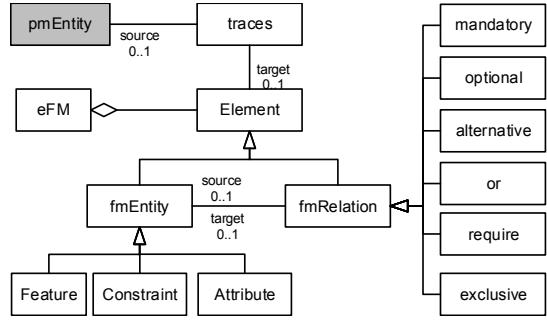


Fig. 5. The meta-model of Evolutionary Feature Model.

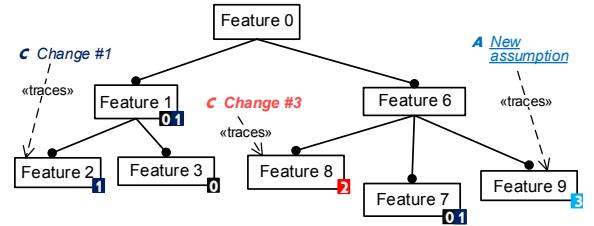


Fig. 6. An example of Evolutionary Feature Model (*eFM*).

Attribute of a feature, or a complex *Constraint*. A *Relation* denotes one of basic relations discussed in Section II. A *traces* relation connecting a *pmEntity* of an *ePM* to an *fmEntity* to denote the situation that an *fmEntity* belongs to.

The feature model of a given situation in *ePM* could be generated easily from the *eFM* by filtering out all entities and their related relations which are explicitly marked as belonging to other situations, but not the given situation.

Example 2. Fig. 6 exemplifies an *eFM* model which incorporates all design changes due to external changes presented in the corresponding *ePM* in Fig. 4. Dashed arrows with label *traces* represent the link from external changes in *ePM* to elements in *eFM*. Elements belonging to some (not all) situations are decorated with small solid boxes with situation's index inside, *i.e.*, **0-3** stand for $F_0 - F_3$ in all milestones. Other elements without such decoration belong to all situations. The F_0 will include features 0, 1, 3, 6, and 7; and F_1 will include features 0, 1, 2, 6, 7. ■

Even though an *eFM* will increase the complexity of the feature model, we still adopt it due to its following advantages:

- Intuitively, the *before* and *after* feature models share a large portion of the features. These shared features need to be replicated among models. Any modifications to the shared part consequently need to be replicated in all models, which may be costly and error-prone. The *eFM* merges *before* and *after* models in a single one, thus it avoids this replication problem.
- The *eFM* captures all design changes in a single model which facilitates a global view of the evolution, and the difference of design changes among situations.

V. DECISION SUPPORT ON FEATURE MODEL EVOLUTION

This section describes the analyses that exploit *ePM* and *eFM* to provide more information about the survivability and cost of repair of a configuration.

A. Survivability Analysis

The survivability analysis measures whether a configuration could be still operational without any modifications despite of the occurrence of evolution. The analysis computes the following quantitative metrics: Max Belief, Residual Disbelief, and Max Disbelief. The first two metrics are based on our previous work on requirements evolution [30] in order to assess a configuration at a point in time.

- *Max Belief (MB)* is a time series measuring the maximum belief that a configuration will still be a valid configuration represented by any *after* feature models at certain milestone t within the study period.
- *Residual Disbelief (RD)* is a time series of the complements of total belief that a configuration will still be a valid configuration represented by any *after* feature models at certain milestone t .
- *Max Disbelief (MD)* is a time series measuring the maximum belief that a configuration will not be a valid configuration represented by any *after* feature models at certain milestone t .

To facilitate the formulation of these metrics, we employ the operation *valid* [2] that takes a feature model and a configuration as inputs and returns 1 if the configuration is represented by the feature model, 0 otherwise.

$$\text{valid}(F, C) = \begin{cases} 1 & \text{if } C \text{ is represented by } F \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

We define an operation *validpos* that takes a feature model and a configuration as inputs, and returns a set of evolution possibilities where the given configuration is valid in *after* model evolution:

$$\text{validpos}(F, C) = \left\{ F \xrightarrow{p_i} F_i \in r_o(F) \mid \text{valid}(F_i, C) = 1 \right\} \quad (4)$$

We further define an operation *age*, which takes a feature model and returns the milestone when the feature model supposes to be. The formula of *age* is as follows:

$$\text{age}(F) = \begin{cases} 1 + \text{age}(F') & \text{if } \exists F' : \langle F' \xrightarrow{p} F \rangle \in r_o(F') \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The cumulative values of Max Belief, Residual Disbelief, Max Disbelief at milestone t for a configuration C , given a feature model F , are depicted as follows:

$$MB(t, C|F) = \begin{cases} \text{valid}(F, C) & \text{if } \text{stable}(F, t) \\ \max_{F \xrightarrow{p_i} F_i \in \text{validpos}(F, C)} p_i \cdot MB(t, C|F_i) & \text{otherwise} \end{cases} \quad (6)$$

$$RD(t, C|F) = \begin{cases} 1 - \text{valid}(F, C) & \text{if } \text{stable}(F, t) \\ 1 - \sum_{F \xrightarrow{p_i} F_i \in \text{validpos}(F, C)} p_i \cdot (1 - RD(t, C|F_i)) & \text{otherwise} \end{cases} \quad (7)$$

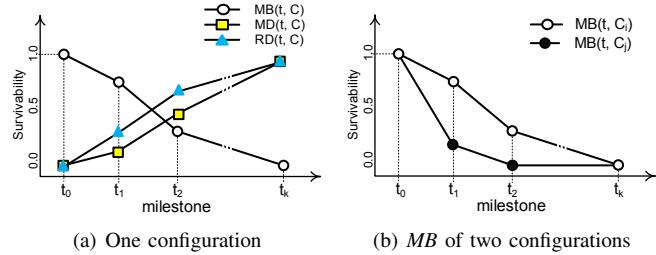


Fig. 7. The survivability diagram.

$$MD(t, C|F) = \begin{cases} \text{valid}(F, C) & \text{if } \text{stable}(F, t) \\ \max_{F \xrightarrow{p_i} F_i \in \text{validpos}(F, C)} p_i \cdot MD(t, C|F_i) & \text{otherwise} \end{cases} \quad (8)$$

where $\text{stable}(F, t) = (r_o(F) = \emptyset \vee \text{age}(F) \geq t)$.

The equation (6) means that, if the given F is stable before milestone t , the validity of the given configuration C within F does not change. As the result, the Max Belief of C in F , captured by as $MB(t, C|F)$, is determined by the validity of C in F . If F evolves, its evolution is denoted in the observable rule $r_o(F)$, the $MB(t, C|F)$ is the maximum belief p_i of all possibilities where C is a valid configuration in their *after* models, multiplied by the Max Belief of C in these *after* models if they continue to evolve. The ideas behind (7), (8) are similarly explained.

Given that F_0 is the original feature model, the time series of Max Belief, Residual Disbelief, and Max Disbelief for a varying t are defined as follows:

$$MB(t, C) = MB(t, C|F_0) \quad (9)$$

$$RD(t, C) = RD(t, C|F_0) \quad (10)$$

$$MD(t, C) = MD(t, C|F_0) \quad (11)$$

These time series could be plotted on a diagram called *Survivability Diagram*, see Fig. 7, showing the development of these series of a configuration over time. In Fig. 7(a) at milestone t_0 , an arbitrary configuration C is valid in F_0 , its $MB(t_0, C)$ is obviously 100%, whereas its $RD(t_0, C)$, and $MD(t_0, C)$ are zero. With subsequent milestones t_i , when evolution occurs the value of $MB(t_i, C)$ might decrease, while $RD(t_i, C)$, and $MD(t_i, C)$ might increase.

The survivability diagram can contain only one or two series of the above metrics of two or more configurations, see Fig. 7(b) for the Max Belief of two configurations C_i and C_j . Such information could provide extra visual information to support the decision making process.

The above metrics can be used to relatively compare among configurations. We compare configurations based on individual metrics. In particular, the comparison between configurations using Max Belief follows the rule: *the higher Max Belief, the better configuration*. Meanwhile, the comparison using *RD* and *MD* is done in the opposite: *the lower Residual Disbelief/Max Disbelief, the better configuration*.

B. Repair Cost Analysis

Few configurations could survive from the beginning to the very end of the study period. Such configurations might require a large investment on features that are only required in later milestones. Hence, decision makers have to consider a trade-off between two strategies: whether to implement all features at the beginning, or postpone some features later. To support such decision, this section provides the analysis on cost of reparation which takes into account the expense to repair an invalid configuration at a certain milestone.

We define two operations, namely `rep cst` and `repair`, that both take a feature model and a configuration as inputs. The former returns a value, called *repair cost*, to represent the minimum cost to make the given configuration become a valid one represented by the given feature model. The latter returns a set of repaired configurations, which have minimal costs.

A straightforward definition of these operators is to enumerate all possible configurations of the input feature model and compute the minimum. Such enumeration has been supported by many studies with automated implementations (see Table 3 in [2]). We also use it because it makes the formal definition easier to grasp. Given a set of features C and a feature model F , let \mathcal{C}_F be the set of all configurations of F . We select configurations C' in \mathcal{C}_F such that the cost to migrate C to C' (e.g., cost to implement new features) is minimized. The formulation of `rep cst` and `repair` is as follows:

$$\text{rep cst}(F, C) = \min \{ \underbrace{\text{cost of adding new features}}_{\text{cost of removing obsolete features}} + \underbrace{\text{cst}'(C \setminus C_i)}_{\text{cost of removing obsolete features}} \mid C_i \in \mathcal{C}_F \} \quad (12)$$

$$\text{repair}(F, C) = \operatorname{argmin}_{C_i \in \mathcal{C}_F} (\text{cst}(C_i \setminus C) + \text{cst}'(C \setminus C_i)) \quad (13)$$

where $\text{cst}(C_i \setminus C)$ denotes the cost of adding new features which are in a valid configuration, but not in the given one; $\text{cst}'(C \setminus C_i)$ is the cost of removing obsolete features in the given configuration, but not in a valid one. For the actual implementation, a more efficient result is obtained by computing directly the minimal number of fixes for the configurations [2]. This can be done with local search algorithms.

The repair cost analysis measures the weighted average reparation costs to repair a configuration due to evolution within the study period. The analysis takes into account all costs to repair a configuration in *after* situations of possibilities in observable rules, and averages them with weights that are equal to the probabilities of the evolution possibilities. The outcome is a time series representing the cumulative reparation cost of a configuration at every milestone.

Let F be a feature model, we define the Cost-of-Reparation (CoR) of a configuration C at a certain milestone t as follows:

$$CoR(t, C|F) = \begin{cases} \text{rep cst}(F, C) & \text{if stable}(F, t) \\ CoR'(t, C|F) & \text{otherwise} \end{cases} \quad (14)$$

$$CoR'(t, C|F) = \sum_{\substack{p_i \\ F \xrightarrow{p_i} F_i \in r_o(F)}} p_i \cdot \min_{C_j \in \text{repair}(F_i, C)} (\text{rep cst}(F_i, C) + CoR(t, C \cup C_j|F_i))$$

Given that F_0 is the original feature model, the Cost-of-Reparation of a configuration C at a certain milestone t is defined as follows:

$$CoR(t, C) = CoR(t, C|F_0) \quad (15)$$

By considering both cost-of-reparation and the initial implementation cost of each configuration, we can calculate the cumulated implementation cost of a configuration at each milestone as follows:

$$Cost(t, C) = \begin{cases} \text{initial implementation cost} & \text{if } t = 0 \\ Cost(t - 1, C) + CoR(t, C) & \text{otherwise} \end{cases} \quad (16)$$

The plot of cumulated implementation cost is referred to as *Cost Diagram* which shows the development of the implementation cost of configuration. We do not give example of Cost diagram here but in the next section. This diagram and *Survivability Diagram* previously discussed could be useful hints to facilitate the configuration selection.

VI. APPLICATION OF THE PROPOSED APPROACH

This section exemplifies the proposed approach in the Smart Grid scenario taken from the NESSoS European project. The evolution described in this section is real, and is taken from the Smart Grid evolution road maps [21], [22]. The evolution probabilities are invented for the illustrative purpose.

A. The Smart Grid Scenario

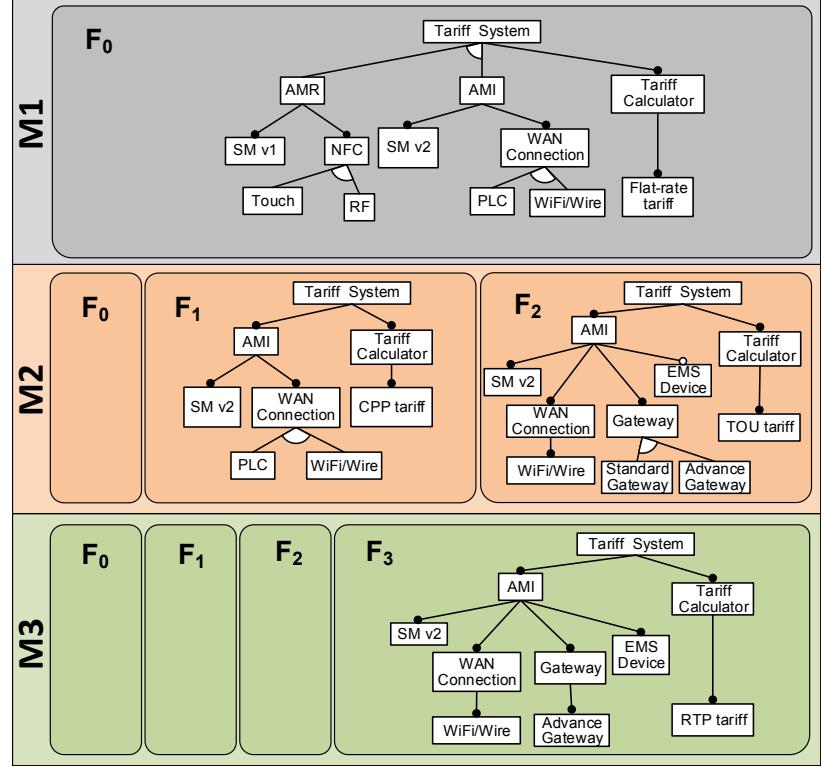
According to the European Technology Platform [10], the Smart Grid is “an electricity network that can intelligently integrate the actions of all users connected to it—generators, consumers and those that do both—in order to efficiently deliver sustainable, economic and secure electricity supplies”. A Smart Grid uses ICT technologies and operates in parallel with an electric power grid to optimize the transmission and distribution of electricity. According to [15], the evolution of Smart Grid has three milestones. The future infrastructure will be based on one-way Automated Meter Reading (AMR) technology, which allows utilities to remotely collect meter data. Next, AMR is replaced or supplemented by the Automated Metering Infrastructure (AMI) to enable utilities to remotely change parameters inside meters, or turn on/off switches. Finally, customers are allowed to actively choose appropriate price options, or utilities that best suit their needs. Different European countries have started to roll out the different capabilities of the Smart Grid infrastructure. For example Britain is currently discussing the roll out of AMR technology. Italy has already rolled out AMR and the authors own buildings have operational elements of AMI allowing utilities to switch off electricity if consumptions exceed a contractual threshold by 20%.

This scenario is very appropriate to capture software product lines: software for different components of the Smart Grid must be customized and shipped based on present needs (e.g., a metering software using 3G connection features versus other network features) but be able to cope with future changes.

Fig. 8. The evolution of the tariff system in the Smart Grid scenario.

To save space, we do not repeat the feature models F_0, F_1, F_2 in M2, M3. Instead, we use round rectangles with label on top to represent the existence of these models.

Acronyms in the diagram: AMR—Automated Meter Reading, AMI—Automated Metering Infrastructure, SM v1—Smart Meter version 1, SM v2—Smart Meter version 2, NFC—Near Field Connection, EMS—Energy Management System, RF—Radio Frequency, PLC—Power Line Connection, WAN—Wide Area Network, TOU—Time-of-Use, CPP—Critical-Peak-Period, RTP—Real-time Period.



Here we focus on the evolution of the tariff system that calculates electric invoices based on collected meter data. The time frame is divided into three milestones: M1, M2, and M3. The evolution of the feature model representing the tariff system is exhibited in Fig. 8, which is described below.

- Milestone M1.** Customers could only choose a utility at the beginning. The meter data will be monthly sent to the utility automatically. The flat-rate tariff is applied, which mean the same price applied for each unit of consumption. For this purpose, either AMR or AMI could be deployed. The AMR requires smart meter with near-field communication technology such as Touch, or Radio Frequency (RF) to send meter data. The AMI requires advanced smart meter which uses either Power Line Connection (PLC) or other Wide Area Network (WAN) connection to send data. The feature model represents this milestone is F_0 .
- Milestone M2.** At this milestone, the tariff system could stay the same as M1 *i.e.*, F_0 . It could also evolve into F_1 . Particularly, more fine-grain tariff could be applied, *i.e.*, Critical-Peak-Period (CPP) where the higher peak-price is restricted to a small number of peak-hours per year, and much higher price for other peak hours. For this purpose, more meter data will be sent to utility in fine-grain intervals. This requires some functionalities of the AMI infrastructure. F_0 could evolve into F_2 which is very similar to F_1 , except a new security requirement to protect customer data arises, and Time-of-Use (TOU) tariff is applied. In this tariff, prices vary between peak-

and off-peak hours. The security requirement requires a home Gateway system to be deployed.

- Milestone M3.** Similarly, all previous feature models $F_0—F_2$ have some chances to be the case in this milestone. Besides, the tariff system could evolve from either F_1 or F_2 into F_3 . In F_3 , there are more utilities in the market, enabling customers to actively choose the utilities as well as different payment options. The tariff calculation is now based on real-time usage of meter data.

B. Construct the ePM and the eFM

Fig. 9 presents the *ePM* of the tariff system of the Smart Grid scenario. Similar to Fig. 4, situations are illustrated as folder shapes where situation names are on the top, followed by the milestones. The propositions are located inside each situation.

The observable rule of the original situation F_0 at milestone M1 is described as follows:

$$r_o(F_0 @ M1) = \left\{ F_0 @ M1 \xrightarrow{0.3} F_0 @ M2, F_0 @ M1 \xrightarrow{0.4} F_1 @ M2, F_0 @ M1 \xrightarrow{0.3} F_2 @ M2 \right\}$$

Fig. 10 shows the *eFM* model of the tariff system which incorporates all design changes due to external changes presented in the corresponding *ePM* in Fig. 9.

C. Perform Survivability Analysis

Table I reports different configurations of the Smart Grid tariff system. Due to the lack of space, we do not report parent features in a configuration. They could be inferred by their

Fig. 9. The *ePM* of the tariff system of the Smart Grid.

Situations are illustrated as folder shapes where situation names F_i followed by the milestones $@M_i$. The situation's propositions are located inside and denoted as **F** – Fact, **A** – Assumption, **C** – Change. An evolution possibility is depicted by an arrow, with probability attached, to connect a *before* situation to an *after* situation.

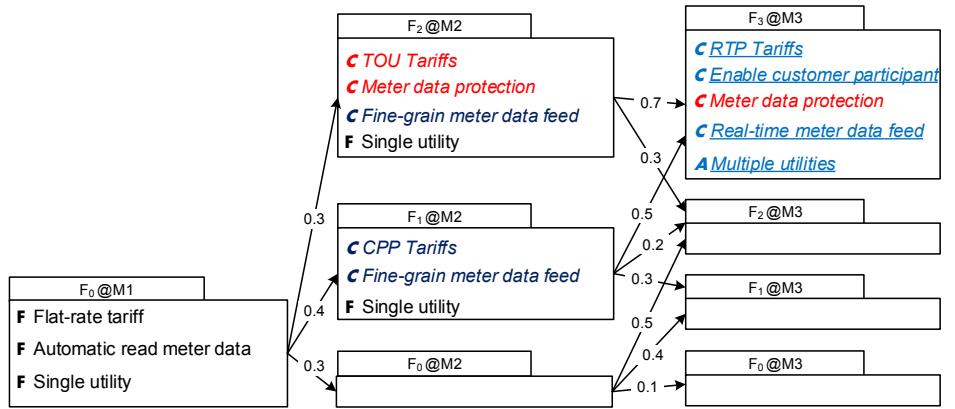


Fig. 10. The *eFM* of the tariff system of the Smart Grid.

Dashed arrows with label *traces* represent the link from external changes in *ePM* to elements in *eFM*. Elements that belong to some (not all) situations are decorated with small solid boxes with situation's index inside, *i.e.*, **0-3** stand for $F_0 - F_3$ in all milestones. Other elements without such decoration belong to all situations.

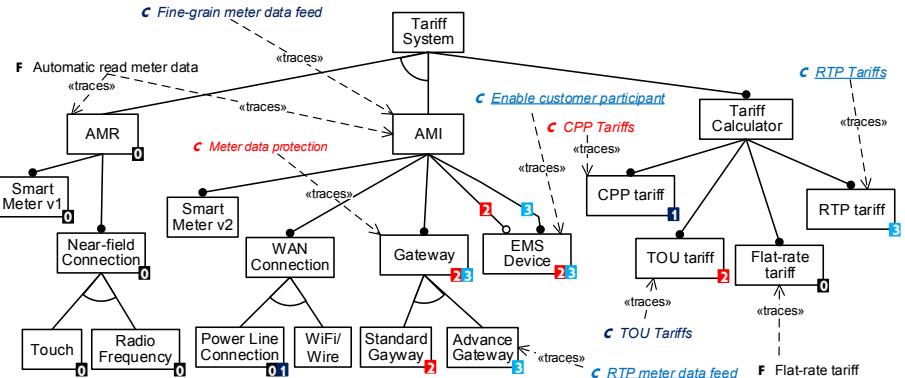


Table I
SURVIVABILITY METRICS FOR THE CONFIGURATIONS OF THE SMART GRID TARIFF SYSTEM.

| Configuration (*) | Valid in | | | M2 | | | M3 | | | |
|---|----------|-------|-------|-------|-------------|-------------|------|-------------|-------------|------|
| | F_0 | F_1 | F_2 | F_3 | MB | RD | MD | MB | RD | MD |
| C1 {SMv1,RF,Flat-rate tariff} | ✓ | | | | 0.30 | 0.70 | 0.40 | 0.03 | 0.97 | 0.21 |
| C2 {SMv2,Wifi,Flat-rate tariff} | ✓ | | | | 0.30 | 0.70 | 0.40 | 0.03 | 0.97 | 0.21 |
| C3 {SMv2,WiFi,CPP tariff,Flat-rate tariff} | ✓ | ✓ | | | 0.40 | 0.30 | 0.30 | 0.12 | 0.73 | 0.21 |
| C4 {SMv2,WiFi,Standard Gateway,TOU tariff, Flat-rate tariff} | ✓ | ✓ | | | 0.30 | 0.40 | 0.40 | 0.15 | 0.73 | 0.21 |
| C5 {SMv2,WiFi,Standard Gateway,TOU tariff, CPP tariff, Flat-rate tariff} | ✓ | ✓ | ✓ | | 0.40 | 0.00 | 0.00 | 0.15 | 0.41 | 0.21 |
| C6 {SMv2,WiFi,Advance Gateway,EMS,RTP tariff,TOU tariff, CPP tariff,Flat-rate tariff} | ✓ | ✓ | ✓ | ✓ | 0.40 | 0.00 | 0.00 | 0.21 | 0.00 | 0.00 |

(*): To save space we do not show parent features in the configurations.

selected children *e.g.*, the configuration $C1$ in Table I is fully reported as {Tariff System, AMR, SMv1, NFC, RF, Tariff Calculator, Flat-rate tariff}. For each configuration, the table reports situations where it is valid, and its survivability values at milestone M2 and M3. The survivability metrics at M1 are the same for all $C1-C6$, *i.e.*, $MB = 100\%$ and $RD = MD = 0\%$, and are not reported. The survivability diagrams of these configurations of Smart Grid are reported in Fig. 11.

The survivability analysis shows that both $C5$ and $C6$ are two winners at milestone M2 because they both have the highest MB , and the lowest RD and MD . In a longer term – milestone M3, $C6$ is the only winner. For other configurations, $C3$ is better than $C4$ at M2, but it is worse than $C4$ at M3. Both $C1$ and $C2$ are equivalent.

D. Perform Repair Cost Analysis

To illustrate the cost calculation, we simplify the cost of each feature as 1 unit; and assume the cost of removing obsoleted features while repairing a configuration is zero. For actual costs and more detailed information, one can consult [13] and a number of websites¹. We assume an extra 20% of costs for each late implementation of a feature on a deployed system.

Fig. 12 reports the cost diagram for the configurations listed in Table I. This figure, instead of reporting absolute numbers, shows the normalized value to the baseline, which is the smallest value of configurations' costs at the first milestone. In this scenario, the cost of $C1$ (or $C2$) is chosen as the baseline.

¹www.smartgrids.eu, www.netw.doe.gov, www.supergen-networks.org.uk

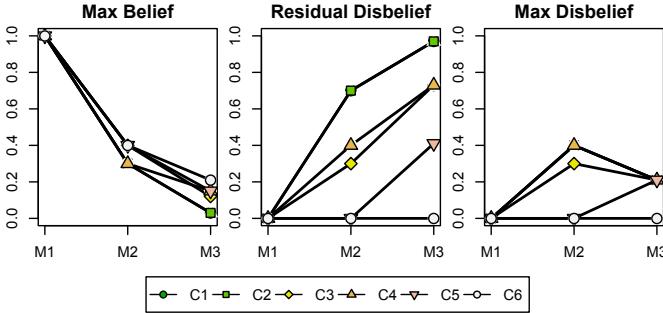
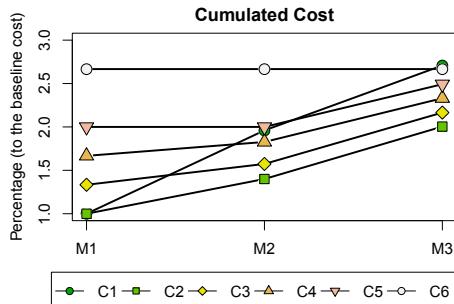


Fig. 11. Survivability diagram for Smart Grid configurations



(a) Cumulated cost diagram

| Configuration | M1 | | | M2 | | | M3 | | |
|---------------|-------------|------|-------------|------|-------------|-----|------|-----|------|
| | Cost | CoR | Cost | CoR | Cost | CoR | Cost | CoR | Cost |
| C1 | 1.00 | 0.96 | 1.96 | 0.75 | 2.71 | | | | |
| C2 | 1.00 | 0.40 | 1.40 | 0.60 | 2.00 | | | | |
| C3 | 1.33 | 0.24 | 1.57 | 0.59 | 2.17 | | | | |
| C4 | 1.67 | 0.16 | 1.83 | 0.50 | 2.33 | | | | |
| C5 | 2.00 | 0.00 | 2.00 | 0.49 | 2.49 | | | | |
| C6 | 2.67 | 0.00 | 2.67 | 0.00 | 2.67 | | | | |

(b) Data table

All costs are normalized to the baseline value, which is the smallest cost of all configurations at the first milestone M1.

Fig. 12. The cost diagram for configurations in Table I.

The medium expensive configuration is C_3 whose cost is 1.33 times higher than C_1 's; and the most expensive configuration is C_6 whose cost is 2.67 times higher than C_1 's.

The diagram shows that universal configurations which try to address all possible evolution in advance are not necessary good choices. Such configurations are C_5 and C_6 (see Table I). These configurations are winners with respect to the survivability analysis in Section VI-C because they anticipate more potential changes in advance than others. The costs for these configurations are always among the highest. In contrast, a naive configuration *i.e.*, C_1 which does not take into account evolution may cost less at the beginning, but might turn to be the most expensive at the end. The configuration C_2 is the most appropriate choice because it has the smallest pace of cost development during the entire study period.

VII. DISCUSSION

Applicability. The applicability of the proposed approach depends on the ability to elicit potential evolution as well as evolution possibilities. Both could be elicited with the aid of domain knowledge. The Smart Grid scenario, and our previous work [30] are examples where evolution could be anticipated. For eliciting probabilities, we can employ a classical approach by Boehm [5] in software risk estimation; a more recent survey can be found in Khan *et al.* [18]. Different approaches can be used to elicit such estimates like use case diagrams [24], lists and questionnaires. Different opinions collected from stakeholders can be combined using ranking [3], analytical hierarchy processes [16], or cumulative sorting [4].

Our approach lacks an empirical validation with external experts. There are two ways of doing it. At first, one could analyze an existing, possibly open-source, system for historical data, then asks developers of the system to assess their decisions in the past. A second alternative is to engage a group of experts and ask their opinion on the input parameters such as probabilities. Then the validity of the prediction results could be validated by them. We leave these for future work.

Scalability. The scalability of the approach is in two folds: the ability to capture and manage big feature models, and the ability to perform reasoning on big feature models. The former fold is a pure technical problem which mostly depends on capability the GUI tool to divide a big model into several diagrams, such as our previous work to manage large model in requirements evolution [31]. The latter fold concerns the scalability of the proposed analyses.

The scalability of the survivability analysis mostly relies on the operator **valid** which has been implemented efficiently in past studies [32], [11].

Similarly, the scalability of the repair cost analysis relies on two operators **repair** and **repCost**. From hypergraph theory [1] we know that polynomial time algorithms exist for MB or alternatively for MD because they are monotone functions. However, this is not true for RD which may require exponential time. So, an alternative would be to approximate the results by using only MB (or MD). The naive implementations for these operators, see (13) and (12), require to enumerate all possible configurations of a feature model. This could be computationally expensive, especially when a feature model consists of a huge amount of features. More efficient approaches could be obtained by adopting artificial intelligence planning techniques. Two promising approaches were proposed by Soltani *et al.* [27] and Ernst *et al.* [9]. The former generates a configuration for a feature model given a set of core features, and set of non-functional preferences and constraints (*e.g.*, minimize of implementation cost). The latter studied a class of algorithms using AI Trust Maintenance Systems to find new configurations that use as much as possible of the old configuration, and minimize the number of new features that need to be implemented.

VIII. CONCLUSION

In this work we have addressed the challenge of dealing with feature model evolution. We have proposed a modeling approach to capture the uncertainty of feature model evolution in terms of two models: Evolution Possibility Model and Evolutionary Feature Model. The former captures different possibilities that a feature model could evolve. The latter captures all changes made to the original feature model due to evolution. The proposed approach allows users to study the continuous evolution of a feature model in several milestones.

We also develop two analyses that exploit the uncertainty of evolution. The first analysis aims to study to what extent a configuration (*i.e.*, set of features to implement) could be resilient to the evolution within the study period. The second analysis aims to understand the development of reparation cost to repair a configuration which is invalid due to evolution. With these analyses, we aim to support the decision makers in selecting an ‘optimal’ configuration regards to the evolution.

ACKNOWLEDGEMENT

This work is partly supported by EU-FP7-IST-NoENESSOS, and EMFASE. We would like to thank the anonymous reviewers for useful comments.

REFERENCES

- [1] G. Ausiello, P. Franciosa, and D. Frigioni. Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In *Theoretical Computer Science*, volume 2202 of *LNCS*, pages 312–328. Springer, 2001.
- [2] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated Analysis of Feature Models 20 Years Later: A Literature Review. *Information Systems*, 35(6):615–636, 2010.
- [3] P. Berander and A. Andrews. Requirements prioritization. In *Engineering and Managing Software Requirements*, pages 69–94. Springer, 2005.
- [4] P. Berander and M. Svhahnberg. Evaluating two ways of calculating priorities in requirements hierarchies - an experiment on hierarchical cumulative voting. *Journal of Systems and Software*, 82(5):836–850, 2009.
- [5] B. W. Boehm. Software risk management: Principles and practices. *IEEE Software*, 8(1):32–41, 1991.
- [6] J. Bosch. Maturity and evolution in software product lines: Approaches, artefacts and organization. In *Software Product Lines*, volume 2379 of *LNCS*, pages 257–271. Springer, 2002.
- [7] G. Botterweck, A. Pleuss, D. Dhungana, A. Polzer, and S. Kowalewski. EvoFM: Feature-driven Planning of Product-line Evolution. In *Proc. of Product Line Approaches in Software Engineering*, pages 24–31, 2010.
- [8] B. H. Cheng, R. De Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, et al. Software engineering for self-adaptive systems: A research roadmap. In *Software engineering for self-adaptive systems*, volume 5525 of *LNCS*, pages 1–26. Springer, 2009.
- [9] N. A. Ernst, A. Borgida, and I. Jureta. Finding incremental solutions for evolving requirements. In *Proc. of the 19th IEEE Intl. Requirements Engineering Conference (RE)*, pages 15–24, 2011.
- [10] European Commission. European smart grids technology platform: vision and strategy for Europe’s electricity. Technical report, 2006.
- [11] D. Fernandez-Amoros, R. H. Gil, and J. C. Somolinos. Inferring information from feature diagrams to product line economic models. In *Proc. of 13th Intl. Software Product Line Conference (SPLC)*, pages 41–50, 2009.
- [12] N. Gamez and L. Fuentes. Software product line evolution with cardinality-based feature models. In *Top Productivity through Software Reuse*, volume 6727, pages 102–118. 2011.
- [13] C. W. Gellings. *The Smart Grid: Enabling Energy Efficiency and Demand Response*. The Fairmont Press, Inc., 2009.
- [14] J. Guo, Y. Wang, P. Trinidad, and D. Benavides. Consistency maintenance for evolving feature models. *Expert Systems with Applications*, 39(5):4987 – 4998, 2012.
- [15] F. Hassan. The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1):18–28, 2010.
- [16] S.-M. Huang, I.-C. Chang, S.-H. Li, and M.-T. Lin. Assessing risk in erp projects: identify and prioritize the factors. *Industrial Management and Data Systems*, 104(8):681–688, 2004.
- [17] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-Oriented domain analysis (FODA) feasibility study. Technical report, CMU/SEI-90-TR-21, Carnegie Mellon University, 1990.
- [18] M. A. Khan, S. Khan, and M. Sadiq. Systematic review of software risk assessment and estimation models. *Intl. Journal of Engineering and Advanced Technology (IJEAT)*, 1(4):298–305, 2012.
- [19] W. Lam and M. Loomes. Requirements evolution in the midst of environmental change: a managed approach. In *Proc. of the 2nd Euromicro Conference on Software Maintenance and Reengineering (CSMR)*, pages 121–127, 1998.
- [20] J. Momoh. *Smart Grid: Fundamentals of Design and Analysis*. Wiley-IEEE Press, 2012.
- [21] National Energy Technology Laboratory. A vision for the smart grid. Technical report, 2009.
- [22] NESSoS project. Selection and documentation of the two major application case studies. NESSoS Deliverable 11.2, 2011.
- [23] A. Pleuss, G. Botterweck, D. Dhungana, A. Polzer, and S. Kowalewski. Model-driven support for product line evolution on feature level. *Journal of Systems and Software*, 85(10):2261–2274, 2012.
- [24] M. Sadiq, M. Rahmani, M. Ahmad, and S. Jung. Software risk assessment and evaluation process (SRAEP) using model based approach. In *Proc. of Intl. Conference on Networking and Information Technology (ICNIT)*, pages 171–177, 2010.
- [25] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein. Requirements-aware systems: A research agenda for re for self-adaptive systems. In *Proc. of 18th IEEE Intl. Requirements Engineering Conference (RE)*, pages 95–103, 2010.
- [26] S. Schach and A. Tomer. Development/maintenance/reuse: Software evolution in product lines. In *Software Product Lines*, volume 576 of *Engineering and Computer Science*, pages 437–450. Springer, 2000.
- [27] S. Soltani, M. Asadi, D. Gašević, M. Hatala, and E. Bagheri. Automated planning for feature model configuration based on functional and non-functional requirements. In *Proc. of 16th Intl. Software Product Line Conference (SPLC)*, pages 56–65, 2012.
- [28] T. Thum, D. Batory, and C. Kastner. Reasoning about edits to feature models. In *Proc. of the 31st Intl. Conference on Software Engineering (ICSE)*, pages 254–264, 2009.
- [29] L. M. S. Tran. Early dealing with evolving risks in long-life evolving software systems. In *Proc. of Advanced Information Systems Engineering Workshops – CAiSE Workshops*, pages 518–523, 2013.
- [30] L. M. S. Tran and F. Massacci. Dealing with known unknowns: Towards a game-theoretic foundation for software requirement evolution. In *Proc. of 23rd Intl. Conference on Advanced Information Systems Engineering (CAiSE)*, pages 62–76, 2011.
- [31] L. M. S. Tran and F. Massacci. Unicorn: A tool for modeling and reasoning on the uncertainty of requirements evolution. In *Proc. of CAiSE Forum*, pages 161–168, 2013.
- [32] J. White, D. Schmidt, D. Benavides, P. Trinidad, and A. Ruiz-Cortes. Automated diagnosis of product-line configuration errors in feature models. In *Proc. of 12th Intl. Software Product Line Conference (SPLC)*, pages 225–234, 2008.
- [33] H. Ye and W. Zhang. Formal definition of feature models to support software product line evolution. In *Proc. of Software Engineering Research and Practice*, pages 349–355, 2008.

Maintaining Requirements for Long-Living Software Systems by Incorporating Security Knowledge

Stefan Gärtner*, Thomas Ruhroth†, Jens Bürger†, Kurt Schneider*, and Jan Jürjens†

* Software Engineering Group, Leibniz Universität Hannover, Germany

{stefan.gaertner,kurt.schneider}@inf.uni-hannover.de

† Chair of Software Engineering, TU Dortmund, Germany

{thomas.ruhroth,jens.buerger,jan.jurjens}@cs.tu-dortmund.de

Abstract—Security is an increasingly important quality facet in modern information systems and needs to be retained. Due to a constantly changing environment, long-living software systems “age” not by wearing out, but by failing to keep up-to-date with their environment. The problem is that requirements engineers usually do not have a complete overview of the security-related knowledge necessary to retain security of long-living software systems. This includes security standards, principles and guidelines as well as reported security incidents. In this paper, we focus on the identification of known vulnerabilities (and their variations) in natural-language requirements by leveraging security knowledge. For this purpose, we present an integrative security knowledge model and a heuristic method to detect vulnerabilities in requirements based on reported security incidents. To support knowledge evolution, we further propose a method based on natural language analysis to refine and to adapt security knowledge. Our evaluation indicates that the proposed assessment approach detects vulnerable requirements more reliable than other methods (Bayes, SVM, k-NN). Thus, requirements engineers can react faster and more effectively to a changing environment that has an impact on the desired security level of the information system.

Index Terms—Security requirements, Heuristics, Requirements analysis, Software evolution, Knowledge carrying software

I. INTRODUCTION

Security is an important quality facet in modern information systems and has to be particularly considered when maintaining such systems. But changing environmental conditions as well as growing attacker knowledge can compromise security of long-living information systems, even if the system itself remains unchanged. As a consequence, a system must be continuously protected against security breaches in a proactive manner. Moreover, software systems tend to degrade when they are modified and extended over a long period of time. Adding a new functionality to a system or modifying one may introduce security-related data and features which remain uninspected. Furthermore, security incidents which occur over time may lead to changes of various parts of the system. These changes may introduce serious vulnerabilities to the regarded information system. The challenge is to maintain an appropriate level of security over a long period of time.

To address this problem, we proposed the SecVolution approach in [34], which continuously monitors security knowl-

edge and its evolution in order to determine required changes to the information system. It therefore combines heuristics for identifying security issues in requirements with a security preservation technique focusing on the system model. They are integrated in a learning cycle supported by knowledge of the environment. The knowledge supports identification of recurring problems in incoming changes. Whenever a problem is detected, a co-evolution mechanism provides semiautomatic recovery of the desired security level of the system model. By applying our SecVolution approach, a long-living system is wrapped into a layer of relevant security knowledge.

Assessing requirements of an information system with respect to security is worthwhile because software design is directly affected by requirements. Considering security later or not at all may result in an insecure system that leads to higher maintenance costs. In this paper, we therefore partly apply our SecVolution approach to support security assessment of requirements. Regarding long-living information systems, security knowledge and its evolution with respect to given requirements is focused. Thus, we propose an integrative security knowledge model to enable the reuse of knowledge. Furthermore, we present a heuristic method that uses this knowledge to support identification of security vulnerabilities in natural language requirements.

The goal of our research is not to detect or even to forecast new security issues, but to incorporate security knowledge in form of reported security incidents supporting security assessment of requirements. Therefore, we focus on requirements specifications by means of use cases. They support understanding of what the requirements of the system are and how the system works.

The remainder of this paper is structured as follows. In Sec. II, we sketch our approach and introduce our research questions. In Sec. III, we review existing security taxonomies and ontologies to identify relevant concepts of security knowledge. Based on the security knowledge, a security requirements assessment built upon heuristics is introduced in Sec. IV and details are presented in Sec. V. In order to refine and to extend the security knowledge, we address the extraction of additional knowledge based on heuristic findings in Sec. VI.

In Sec. VII, the proposed assessment approach is evaluated by using the iTrust case study. Related research is presented in Sec. VIII. Finally, we conclude our work and outline future research in Sec. IX.

II. RESEARCH OBJECTIVE

This paper addresses the management and usage of evolving security knowledge in requirements engineering. The benefit of our approach is to support requirements engineers, so that they are able to invest their efforts into prevention and security improvement instead of finding known security issues in lengthy specification documents.

To achieve this objective, we propose an approach as presented in Fig. 1. Our approach consists of two parts: (1) Security Assessment and (2) Security Knowledge Extraction. In the first part, a given specification is assessed using heuristics and security knowledge. Heuristic findings are passed to the second part where a requirements engineer examines them and extracts additional knowledge as well as security requirements. Additional knowledge is used to adapt or to refine applied security knowledge whereas security requirements are passed to subsequent development activities. Regarding our approach, we have to face three research questions as depicted in Fig. 1.

Security knowledge is spread among information sources of very different kind (e.g. laws and regulations, attackers and white hats, incidents, developers, other stakeholders). Even security experts cannot entirely overview the relevant knowledge. Thus, we need to know (**RQ1:**) *How to organize security knowledge in a way that it can be used for assessing requirements of a long-living software system?*

To offer useful support for requirements engineers based on reported security incidents, we need to know (**RQ2:**) *How can requirements engineers identify security-critical issues in natural language requirements semiautomatically?* Here, we focus on use cases which need to be transformed into an appropriate analysis model applying natural language processing. After identifying security issue candidates, requirements engineers need to spot the candidates which really threaten the system.

The core concept of our approach is to use security knowledge modeled beforehand. However, modeling a huge amount of knowledge is a labor-intensive task. This leads to (**RQ3:**) *How can requirements engineers be supported to extract proper security knowledge from identified security-critical issues in requirements?* Since the extraction of knowledge

depends on the way the knowledge is organized, this question is related with our first research question.

To illustrate our concepts and to evaluate our approach, we choose the project iTrust [27] as case study. It is a medical application which supports patients to manage their health records as well as medical staff to organize their work. The health care sector is quite complex and security plays an important role. For example, a lot of sensitive patient data have to be stored. The iTrust project was founded at North Carolina State University and is currently maintained by the Realsearch Research Group [27]. It consists of 55 use cases written in natural language (version 23). Based on these requirements, the iTrust system (version 17) has been developed as a web application. Thus, the listed requirements are sufficient to develop a working system.

III. MODELING SECURITY KNOWLEDGE

Security experts' knowledge consists of a mixture of textbook knowledge, security obligations and laws, as well as experiences comprising typical and exceptional cases. Especially for novices, it is difficult to find the relevant information to cope with a particular problem.

In this section, we clarify requirements on security knowledge. Based on this, we review several security taxonomies and ontologies to identify relevant security concepts being appropriate to document relevant security incidents.

A. Requirements on Security Knowledge

Regarding security, we need to deal with the unknown unknowns [26]. This refers to the problem that we cannot say which knowledge will be relevant in the future. Moreover, security knowledge is not necessarily limited and can change rapidly. Even established best practices can become out-dated. For example, a delay after entering a wrong password several times in order to login might be sufficient for local computers, but could facilitate a denial-of-service (DoS) attack on a web-based infrastructure. To cope with these problems, security knowledge needs to be maintained. This means that knowledge elements must be added or modified by humans iteratively. Therefore, it has to be represented in a symbolic manner instead of being modeled by statistical means (e.g. Naive Bayes). To enable semiautomatic security assessment, it needs to be suitable for computer-based processing.

B. Security Ontology and Concepts

In our approach, security knowledge consists of security incidents enriched with additional information retrieved from natural language documents such as security guidelines and obligations. An incident is an attempt to violate security policies or to gain unauthorized access to data [15].

In recent years, various taxonomies and ontologies for modeling incident-centric security knowledge have been proposed. Since security knowledge is widely used, we focused on publications from the area of threat modeling, risk analysis, computer and network security, software vulnerabilities and information security management. Moreover, we consider

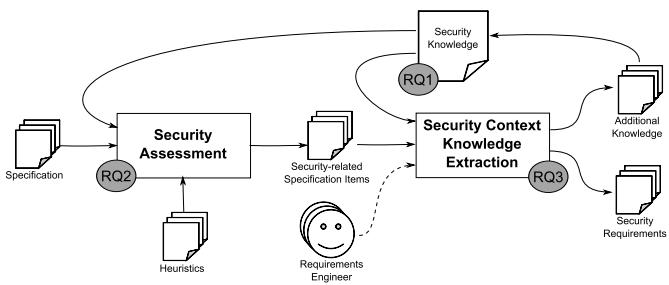


Fig. 1. Overview of the proposed approach including research questions.

TABLE I
PRIMARY STUDIES AND THEIR PRINCIPAL SECURITY CONCEPTS

| Publication | Principal Security Concepts |
|-------------------------|--|
| Howard et al. [18] | Action, Target, Access, Tool, Vulnerability, Result, Objective, Attacker |
| Jung et al. [21] | Asset, Vulnerability, Threat, Security Control, Risk Probability, Asset Value, Impact, EC environment |
| Mouratidis et al. [29] | Constraints, Secure Entity (goals, tasks, resources), Secure Dependency |
| Undercoffer et al. [42] | Attack, System Component, Input, Consequence, Means, Location |
| Alvarez et al. [3] | Entry Point, Vulnerability, Service, Action, Input Length, HTTP Headers, HTTP Verb, Target, Scope, Privileges |
| Swiderski et al. [40] | Asset, Entry Point, Trust Level, Attack, Attacker, Vulnerability, Countermeasure |
| Herzog et al. [16] | Asset, Threat, Vulnerability, Countermeasure |
| Tsoumas et al. [41] | Asset, Risk, Threat, Attack, Threat Agent, Vulnerability, Impact, Countermeasure, Controls, Security Policy, Stakeholder |
| Karyda et al. [22] | Asset, Countermeasure, Objective, Person, Threat |
| Barnum et al. [5] | Vulnerability, Weakness, Method of Attack, Attack Consequence, Attacker Skill, Solution and Mitigation, Resource, Context |
| Fenz et al. [12] | Asset, Organization, Security Attribute, Threat, Threat Source, Threat Origin, Vulnerability, Control, Severity Scale |
| Elahi et al. [10] | Vulnerability, Effect, Attack, Security Impact, Malicious Goal, Attacker, Countermeasure, Malicious Action, Component, Actor |
| Simmons et al. [38] | Attack Vector, Operational Impact, Defense, Informational Impact, Target (Network, Application, etc.) |
| Guo et al. [13] | Attack, Countermeasure, Consequence, Attacker, Vulnerability, IT Product |
| Miede et al. [28] | Attack, Countermeasure, Asset, Vulnerability, Threat, Security Goal |
| Eichler [8] | Asset, Threat, Damage Scenario, Protection Requirements, Safeguard, Module |

publications covering information systems, cyber-physical systems, distributed systems and agent-based systems. We ignored earlier taxonomies because they have been considered in recent publications. To cope with security issues in requirements engineering, we rejected publications which mainly focus on technical security aspects of systems (e.g. protocols, encryption algorithms). Identified primary studies and their principal security concepts are presented in Table I. Based on these concepts, we extracted a minimal knowledge structure, which can be found in each of the regarded taxonomies in one or the other way. In Fig. 2, the regarded security concepts and their relationships are presented. Based on the reviewed publications, we define the following concepts to model incidents and illustrate them with examples from the iTrust project:

- An *asset* is an item of interest worth being protected (e.g. password, medical identification number, patient and office visit information).
- *Entry points* define the interfaces to interact with the system. They provide access to assets (e.g. login website, health record, email, input field).
- A *trust level* describes which role has access to an asset using a specific entry point (e.g. patient, health care personnel, administrator).
- *System components* model the regarded system focusing on assets and entry points. This includes hardware as well as software components (e.g. database, logging).

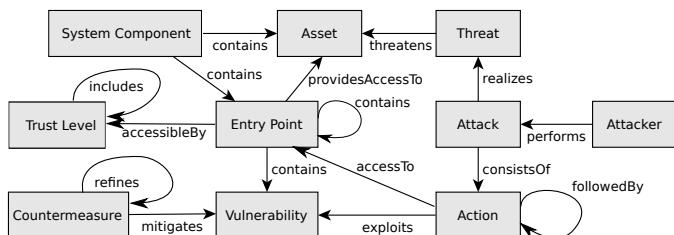


Fig. 2. Overall structure of the security knowledge. High level concepts and their relationships are presented. The relation *followedBy* marked with an asterisk represents an ordered list in a simplified form.

- An *attack* is a sequence of malicious *actions* that are performed by an attacker aiming at assets (e.g. cross-site scripting, denial-of-service attack).
- A *vulnerability* is a system property that facilitates unintended access or modification of assets. It violates an explicit and implicit security policy. Entry points may have or provide access to vulnerabilities (e.g. improper neutralization of input, missing encryption of sensitive data).
- A *threat* is the possibility to perform a successful attack on a specific asset. Successful attacks exploit at least one vulnerability to cause damage (e.g. execute unauthorized code or commands, expose sensitive data).
- A *countermeasure* mitigates a certain threat by fixing the respective vulnerability (e.g. input validation, encryption of sensitive data).

Based on our aforementioned requirements, we decided to use ontologies to organize security knowledge. Moreover, reasoning and ontology query languages allow to have unified access to knowledge. Because security knowledge can be invalidated over time, we need to recognize when contradictory data is included. For this purpose, reasoning techniques help us to find such knowledge.

C. Knowledge Acquisition and Incident Documentation

To support vulnerability prevention for requirements, incidents must be documented and enriched with security knowledge which is used to derive effective mitigation strategies. This knowledge can be retrieved from various sources such as security guidelines and standards, best practices as well as vulnerability catalogs. In our approach, an incident is modeled using corresponding actions which refer to several entry points, assets, and trust levels (see Fig. 2). Moreover, the vulnerability which has been exploited by an action has to be documented. For this purpose, mature incident documentation approaches exist [2]. This is not within the scope of this paper. Instead, we focus on refinement of the gathered incidents in order to improve security assessment of requirements.

IV. SECURITY ASSESSMENT IN REQUIREMENTS ENGINEERING BASED ON HEURISTICS

A security assessment is a systematic technique to analyze the current security status of the system under development [43]. This also includes identifying vulnerabilities in requirements. They are usually written in natural language to facilitate communication among different stakeholders. Assessing these documents with respect to security is a human-intensive and time-consuming task. Moreover, occurred security incidents may reveal new vulnerabilities which have not been considered by the requirements engineer yet. As a consequence, all requirements have to be assessed again taking the discovered weakness into account.

As an example, QR codes, which are used along with mobile devices (e.g. smart phones, tablets), introduced a new attack vector. QR codes are machine-readable barcodes which encode information about an item. To a large extent, they are used on advertising posters and contain links to the vendor's website. As previously reported [19], attackers stick manipulated codes over regular ones linking to malicious websites. People who used these codes without checking the target link infected their mobile devices. Based on this incident report, a requirements engineer needs to identify all use cases of the regarded system where QR codes are used. These findings are then analyzed in order to plan effective mitigation strategies.

This example also illustrates that newly discovered attack techniques and technological progress constitute new challenges. Regarding long-living software systems, the problem becomes even more challenging. In order to decrease the delay between discovering new incidents and assessing the specification, semiautomatic assessment techniques in requirements engineering are desired.

Security assessment of requirements comprises information about the security problem and references to effective mitigation strategies. For this purpose, we developed a heuristic to identify reported vulnerabilities and their variations in natural language requirements based on revealed security incidents.

A. Towards Heuristics in Requirements Engineering

In this paper, a heuristic is defined as an analytical method to assess requirements with respect to security. Heuristics are based on hypotheses and have to cope with incomplete or unknown knowledge. Heuristic findings are suboptimal which means that in many cases they are correct, but there are false positives. The challenge is to develop heuristics that identify all intended situations reliably and, thus, reduce the number of false negatives. This gets even more challenging for long-living systems due to evolving assumptions and hypotheses. As a consequence, an initial heuristic may perform well but gets worse later on.

In our previous work, we tested different mechanisms to establish helpful heuristics for requirements engineering. In our first approach, we attempted informed guessing from experiences or from literature (e.g. no passive voice, not too long sentences) [24]. Based on this, we codified security experiences by statistical means [37]. For this purpose, we

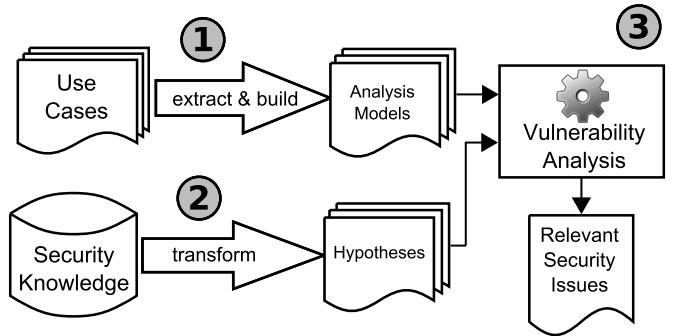


Fig. 3. Security assessment based on modeled security knowledge.

trained a Bayesian classifier with pre-evaluated security material. The classifier performed well for identifying security-relevant requirements automatically [17], [37]. It bridges the gap between security best practices and the lack of security experience among requirements engineers. However, detailed information about the security issue has not been provided by the approach. In this paper, we extend our previous work by using security knowledge to determine detailed security information from use cases. Additionally, we focus on the fact that in long-living systems security knowledge changes.

B. Overview of the Proposed Security Assessment Approach

The aim of our approach is not to replace established and well-suited security assessment methods, but to support them by providing intermediate feedback to the requirements engineer. Therefore, our approach reveals potential security threats and vulnerabilities for given use cases. It takes the scenario description of the use cases as well as security knowledge as input. A brief overview of our assessment approach as illustrated in Fig. 3 is presented in the remaining section. Further details are explained in Sec. V.

In step 1, each use case written in natural language is transformed to a separate analysis model named *security abstraction model*. A security abstraction model consists of an ordered list of steps and represents the usage scenario with respect to security. Each step is described by a set of the following attributes: (1) system components, (2) entry points, (3) assets and (4) trust levels. To infer these attributes from use cases, natural language processing [4] is used. In step 2, hypotheses for the heuristic are built up from security knowledge. Therefore, each documented security incident is also transformed to a separate security abstraction model. The hypotheses are used in step 3 to assess the analysis model extracted from the given use cases. The idea is to detect suspicious sequences within the use cases which have been seen in incidents before. For example, an incident is reported that an attacker obtained sensitive data by intercepting unencrypted emails. Based on this report, a use case is marked as vulnerable if sensitive data (e.g. passwords) are sent via email. On this account, semantic similarity between the steps of the corresponding security abstraction models is derived.

Our approach returns a set of hypotheses which fits best with the given use cases. Hypotheses are used to infer the underlying threat as well as possible countermeasures from the security knowledge. Moreover, additional security information (e.g. security guidelines and standards, vulnerability databases, etc.) is attached to threats and countermeasures which can be used to analyze the vulnerability in detail.

V. DETAILS OF OUR SECURITY ASSESSMENT APPROACH

To compare information from different knowledge sources, it needs to be represented in a uniform manner. For this purpose, we use security abstraction models as a unified representation to encode use cases and security incidents. The proposed security analysis is defined on the basis of the security abstraction models.

Since we consider use cases written in natural language, our approach relies on natural language processing. To cope with ambiguities and imprecision of natural language, we introduce a linguistic approach to determine the semantic similarity between words. Semantic similarity quantifies how much a word *A* has a similar meaning to another word *B* although they are syntactically different [32]. Regarding the health care domain, the term *patient*, which refers to a trust level in iTrust, is semantically similar to the term *sick person* or *hypertensive*. Since we only consider concepts as presented in Sec. III, it is sufficient to focus on nouns. Thus, this enables identifying possible variations of vulnerabilities in requirements and increases the efficiency of our approach.

A. Semantic Similarity Computation for Nouns

The goal is to measure the semantic similarity between nouns based on the structure and content of WordNet. In WordNet nouns are organized in hierarchies [11]. For this purpose, we adapt the approach presented in [20] which is based on the information content of the lowest common subsumer (LCS). The LCS is the concept in a tree-like lexical taxonomy, which has the shortest paths from the two concepts compared. According to WordNet, the term *sick person* is the LCS of the medical terms *diabetic* and *epileptic*. If the information content of the LCS is above a pre-defined threshold, it is too general to represent both concepts. Otherwise, both concepts are semantically similar.

To derive the LCS for the given concepts, the paths are derived by using their hypernyms listed in WordNet. A hypernym is a generic term used to describe a whole class of specific terms [44]. For example, the term *sick person* is the hypernym of the term *patient*. The information content of the LCS is defined as the sum of all concepts which have the LCS in common. For large word taxonomies such as WordNet, this is very CPU-intensive to derive. To overcome this problem, the information content of the LCS is approximated. On this account, for each concept on the paths the number of all direct hyponyms is summed up including all direct hyponyms of the LCS. A hyponym is the specific term used to describe a member of a class [44]. For example, one hyponym of *patient* is the term *hypertensive*.

B. Building the Security Abstraction Models

To assess security as presented above, the usage scenario of use cases as well as security incidents must be transformed to security abstraction models.

1) *Determining Abstractions from Use Cases*: Use cases depict high-level functionality of the system. From a security perspective, a use case specifies which assets are accessed by which actors using certain entry points.

To derive security abstraction models, each use case undergoes a series of steps. Firstly, the description of each scenario step is enriched with syntactic information. Syntactic information specify whether a word is a noun, verb, or adjective. Therefore, a statistical part-of-speech tagger as presented in [35] is applied. Secondly, nouns and compound nouns are extracted and stemmed for each scenario step. Thirdly, extracted nouns are assigned to the attributes (e.g. system component, asset, entry point, trust level) of the respective security abstraction model. For example, the third use case of the iTrust system describes how user authentication is applied. Here, the actor of the use case is a *user* (trust level) who must enter her *medical identification number* and *password* (assets) into the *login form* (entry point) of the application. In iTrust, a user is a patient, a health care personnel, a public health agent, and so on. To assign extracted words to the attributes, the modeled security knowledge serves as a glossary.

Finally, abstracted scenario steps of a use case are arranged in an ordered sequence whereby the precondition is the first element followed by the trigger. The postcondition is put at the end of the sequence. Alternative paths in use cases are considered as separate security abstraction models. Moreover, loops and conditional clauses contained in usage scenarios are currently not evaluated.

2) *Extracting Hypotheses from Security Knowledge*: To derive hypotheses, the overall structure of the proposed security knowledge as presented in Sec. III is used. A hypothesis encodes incidents according to the security abstraction model. For this purpose, threats assigned to an attack (see relation *realizes*) are used to obtain assets which are threatened (see relation *threatens*). Actions of an attack are arranged chronologically using the relation *followedBy*. Each action represents a step in the security abstraction model. Actions refer to entry points which are used to access assets. System components, assets, and trust levels assigned to an entry point are used as attributes respectively. For example, an incident is reported that an attacker obtained passwords by trying words from a dictionary. Therefore, she enters each word into the login form of the application. In the corresponding security abstraction model the asset is set to *password*, the entry point is set to *login form*, and the trust level is set to *unknown* because no further information about the attacker is provided.

C. Measuring Similarity between Security Abstraction Models

To identify vulnerabilities in use cases, the corresponding security abstraction models are evaluated for the occurrence of the steps encoded in the hypotheses. A similar sequence of steps may indicate that the system described by the use case

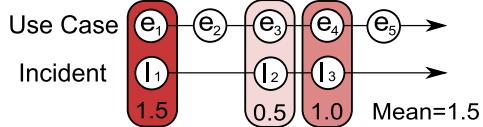


Fig. 4. Alignment between the security abstraction models derived from a use case and a security incident.

is vulnerable with respect to the respective security incident. Concerning this, steps do not have to follow immediately. This means that any two adjacent steps within a hypothesis are separated by an arbitrary number of steps within the security abstraction model of the use case. The maximum match of two sequences is defined as the largest number of similar steps which occur in the same order (sequence alignment) as illustrated in Fig. 4. The numbers in this figure represent the degree of semantic similarity between steps. The corresponding mean value is used to evaluate whether a relevant match has been found.

To derive the alignment in our approach, we utilize the Needleman-Wunsch algorithm [30]. The smallest unit of comparison is a pair of steps, one from each security abstraction model. All possible pair combinations are determined and stored in the corresponding cell of a two-dimensional matrix (dynamic programming). For this purpose, the semantic similarity computation as described in Sec. V-A is used. Every possible comparison can be represented by pathways through the matrix. The maximum match is represented by the pathway for which the mean value of assigned cell values is largest. If the mean value is above a given threshold, a possible vulnerability have been found for the given use case.

Regarding the security abstraction models from our examples in Sec. V-B1 and Sec. V-B2, the asset *password* and the entry point *login form* which have been extracted from the usage scenario are equal to the corresponding attributes of the incident. The trust level *unknown* from the incident contains the trust level *user* from the use case. As a result, the hypothesis matches with the security abstraction model of the use case which is therefore marked as vulnerable.

VI. EXTRACTING SECURITY KNOWLEDGE

In order to develop practical knowledge bases, it is necessary to acquire, understand, encode, and organize human knowledge appropriately. This is important for requirements engineering because it is usually not possible to predefine a definitive body of security knowledge for a certain domain. One reason is that some domains are partially understood and need to be clarified over time. Thus, learning security knowledge helps to improve security of long-living software systems. Learning in our context means (1) acquiring new knowledge or (2) modifying, reinforcing, and refining existing knowledge. The challenge is that we only have a few incidents as problem instances to learn from. For this reason, the aim of this method is to iteratively learn security knowledge.

We define knowledge as the expertise stored in a person's mind which has been obtained by interacting with the person's

environment [36]. To acquire security-relevant knowledge efficiently, we need an elicitation method which is seamlessly integrated into the security analysis workflow. On that account, we propose a method that is built upon the heuristic findings which have been achieved by using our security assessment approach. These findings highlight possible vulnerabilities within the use cases and consist of truly and falsely classified use cases (true and false positives). True positives mainly contain additional knowledge about system components, assets, entry points, and trust levels. However, false positives are used to specify terms for the regarded domain. This means that all meanings of a term, which are not valid in a given domain, are excluded explicitly. To extract knowledge, the requirements engineer has to evaluate all vulnerable use cases. The effort to refine knowledge decreases over time because knowledge of a given system improves.

To extract additional knowledge from true positives, natural language parsing and syntax tree analysis is used. The extraction process is guided by applying the proposed knowledge structure as presented in Sec. III. Input of our method is a set of heuristic findings. As presented in Sec. V-C, they consist of suspicious scenario steps of the use cases which may compromise the security of the system. Each finding undergoes a series of steps. Firstly, the linguistic structure (syntax tree) of the regarded scenario step is determined. Secondly, the syntax tree is analyzed with respect to the attributes (system component, asset, entry point, and trust level) of the corresponding incident. For example, if in the description "The user enters an identification number and a pin." the asset *pin* is identified, it can be implied that *identification number* is also an asset because it has a linguistic dependency to *pin*. Thirdly, the requirements engineer is questioned whether the new knowledge should be added to the knowledge base. Moreover, added knowledge can be enriched by a security expert with additional security information (e.g. security standards and guidelines).

In order to specify terms for a certain domain, false positives are mainly considered. For this purpose, falsely classified scenario steps and their semantic relationships to the respective incident are used. Attributes (system component, asset, entry point, and trust level) of the incident and their corresponding match within the scenario step are evaluated. If the semantic similarity is below a pre-defined threshold the requirements engineer is questioned whether both attributes mean the same in the given domain. Thus, the requirements engineer can actively choose which terms must be excluded from semantic similarity computation as presented in Sec. V-A.

Security knowledge extraction performed by humans as well as refinement of knowledge is also prone to ambiguities. However, we consider the identification and dissemination of vulnerabilities a higher value.

VII. iTRUST CASE STUDY

The aim of this case study is to evaluate whether our approach can support requirements engineers. In our previous work [17], [37], we used a Naive Bayes classifier to derive

whether a requirement is security-relevant or not. Thus, we evaluate the performance of our approach in contrast to Naive Bayes classifiers. Moreover, we also consider Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN) since they perform well in many machine learning tasks.

A. Case Study Design

Ten initial use cases have been selected to set up the security knowledge according to Sec. III. The selection criterion is that at least each use case has a different actor and thus covers a different functionality of the iTrust system. Since iTrust does not have documented security incidents, we prepared misuse cases (MUC) as described in [1]. Therefore, we focused on the technological domain because we are no health care experts.

In order to evaluate whether the performance of our heuristic is improved through knowledge refinement, we performed two iterations. For the first iteration, we selected 35 iTrust use cases. Together with the initial use cases, they have been assessed using our approach. Afterwards, we compared the heuristic findings with our manual assessment which we have done before. Considering true positives and false positives, we refined the security knowledge (see Sec. VI). Based on the refined knowledge, we assessed all use cases in the second iteration.

1) *Misuse Cases*: From analyzing use case 1 (UC1), we obtain misuse case 1 (MUC1). In UC1, an email, which contains the initial password to log into a user account of the iTrust system, is sent to the user which has been created by the health care personnel. In the MUC1, the email is intercepted by an attacker which thus obtains sensitive user information. The attacker uses this information to gain access to the user account. Therefore, hijacking user accounts becomes possible. To mitigate this vulnerability, emails containing sensitive information must be encrypted.

Analyzing UC6 leads to MUC2. In MUC2, the patient view with respect to the address field contains a vulnerability so that cross-site scripting (XSS) attacks become possible. This is one of the most dangerous vulnerabilities in web applications. XSS becomes possible through improper neutralization of input. Attacker can inject malicious browser-executable content into the patient view to steal sensitive information (e.g. medical identification number or password). To mitigate this vulnerability, all use cases need to be identified which render demographic information. Identified use cases and their corresponding code can be checked whether they sanitize input properly.

2) *Security Knowledge*: Based on the initial use cases, an appropriate security ontology is set up as proposed in Sec. III. For this purpose, terms from the iTrust glossary (health care domain) are used as well. Furthermore, system components, assets, trust levels and entry points of the misuse cases are added as listed in Table II.

3) *Naive Bayes, SVM, and k-NN*: To set up Naive Bayes, SVM, and k-NN in the training phase, the initial use cases are labeled with respect to the misuse cases. If a misuse case is related to a use case in any way, the use case is labeled *sec*.

TABLE II
EXTRACT OF SECURITY KNOWLEDGE USED IN iTRUST CASE STUDY.

| Concept | Individuals | |
|-------------|---|-------|
| | MUC 1 | MUC 2 |
| Asset | initial password, security key | |
| Entry Point | email | |
| Trust Level | user | |
| | | |
| Asset | address | |
| Entry Point | address field, health record, view, display | |
| Trust Level | patient, health care personnel (HCP) | |

Otherwise, it is labeled *non-sec*. Here, words of the given use case descriptions are not rated separately since it is usually not easy to decide which words indicate non-security information.

B. Results

The quality of our approach is measured in terms of accuracy (ACC), false positive rate (FPR) and false negative rate (FNR). The accuracy is defined as the degree of correctly assessed use cases with respect to all use cases. The FPR measures the rate of use cases which have been falsely assessed as security-relevant. In contrast, the FNR measures the rate of use cases which have been falsely assessed as not security-relevant. A low FNR is desired since it means that at least all security-relevant use cases have been found correctly.

1) *First Iteration*: In Table III, results of the first iteration with 44 uses cases are presented. Since one incident was not enough to achieve reasonable results, we trained one classifier for both misuse case. As a result, Naive Bayes, SVM, and k-NN have a high FRN which means that a lot of vulnerable use cases were overlooked. Regarding SVM, the FNR is 1.0 which means that no vulnerable use cases were identified. For that reason, we cannot refine knowledge based on heuristic findings. Thus, SVM is not considered in the second iteration.

Our approach identified all five use cases which are related to MUC1 in any way. Thus, the recall of our approach is 1.0. Regarding MUC1, the FNR is 0.0 which mean that no use case was overlooked. Moreover, our approach identified 11 out of 13 use cases which are related to MUC2 in any way. This implies that the recall is around 0.85. However, FNR is 0.15

TABLE III
EVALUATION RESULTS (ACC = ACCURACY, FPR = FALSE POSITIVE RATE, FNR = FALSE NEGATIVE RATE)

| | | ACC | FPR | FNR |
|-----------------------------|---------|------|------|------|
| 1st Iteration (n=44) | | | | |
| Our Approach | MUC 1 | 0.90 | 0.10 | 0.00 |
| | MUC 2 | 0.64 | 0.55 | 0.15 |
| Naive Bayes | MUC 1/2 | 0.61 | 0.00 | 0.89 |
| SVM | MUC 1/2 | 0.57 | 0.00 | 1.00 |
| k-NN | MUC 1/2 | 0.57 | 0.15 | 0.83 |
| 2nd Iteration (n=55) | | | | |
| Our Approach | MUC 1 | 0.98 | 0.00 | 0.14 |
| | MUC 2 | 0.84 | 0.14 | 0.23 |
| Naive Bayes | MUC 1/2 | 0.71 | 0.11 | 0.67 |
| k-NN | MUC 1/2 | 0.76 | 0.00 | 0.68 |

which means that two vulnerable use cases were not identified. The number of false positives is 17 which means that more false positives were found than true positives. Nevertheless, 14 out of 44 use cases can be ignored (true negatives) which reduce the effort for manual assessment by a third. The reason why our approach performed worse in MUC2 is that the asset *address* as well as the entry point *address field* are not very specific for the health care domain and thus are similar to many words semantically. Thus, it is necessary to define the meaning of these terms more precisely.

2) *Knowledge Refinement*: Based on Sec. VI, we extracted additional knowledge from the heuristic findings and thus refined the applied security knowledge. On that account, we considered all use cases which were identified as vulnerable. True positives were used to add additional knowledge. To specify used terms, false positives were mainly considered.

Regarding UC3, the term *security answer* was added as a valuable asset which also should not be sent via email. Moreover, UC25, UC30 and UC40 were used to specify the meaning of the asset *password* with respect to the iTrust system. It was added that *password* is not a *word*, a *number*, or a *diagnosis*.

Analyzing UC32, the entry point *patient report* was added since it contains the patient's address and views it on a webpage. As mentioned above, the asset *address* is not specific for the health care domain. For this purpose, UC3, UC8, UC10, UC32 are used to define that *address* is not an *IP address*, an *email address*, an *access*, a *link*, a *prescription*, or a specific health care *code*.

To re-train the Naive Bayes as well as k-NN classifier, we added the true positives to the training set and labeled them as *sec*. For Naive Bayes two use cases and for k-NN three use cases were considered. Furthermore, we also added the false positive and labeled them as *non-sec*. Thus, four falsely classified use cases were added for k-NN. Naive Bayes produced no false positives.

3) *Second Iteration*: The results of the second iteration with 55 use cases are presented in the lower part of Table III. Regarding Naive Bayes and k-NN, the FNR are lower compared to the first iteration. But only two additional use cases were identified as vulnerable by Naive Bayes. For k-NN only one additional use case was found. As a result, Bayes, SVM, and k-NN are not applicable to identify the majority of vulnerabilities based on incidents in natural language requirements.

Regarding our approach, however, the accuracy is good for both misuse cases. Moreover, the FPR for MUC2 decreased from 0.55 to 0.14. Indeed, no false positives were produced for MUC2. Thus, knowledge refinement has a significant impact on the heuristic findings regarding FPR. The FNR, however, has increased slightly compared to the first iteration. But only true positives, which have been already considered by the requirements engineer in the first iteration, are affected.

C. Discussion

Based on the iTrust case study, we discuss the achieved results with our research questions. To model security inci-

dents which are used in our security assessment approach, a knowledge structure has been presented in Sec. III. Based on this structure, the security abstraction model has been defined. According to **RQ1**, the results of the iTrust case study indicate that the proposed concepts are sufficient to represent use cases and incident in a uniform manner. Moreover, the proposed knowledge structure is small enough to document incident without requiring too much human effort. Only system components, assets, entry points, and trust levels for each step of the incident must be documented.

Answering **RQ2**, identification of vulnerable use cases has been evaluated using the iTrust case study. Since reported incidents are not available for iTrust, we derived two misuse cases from the given requirements. Moreover, we also compared our approach to methods which we developed in our previous research. As a result, Naive Bayes, SVM, and k-NN performed very weak. One reason is that training data is sparse because only a few misuse cases are available to learn from. However, this is common in our usage scenario. In comparison, our security assessment approach achieved better results in terms of accuracy, false positive rate and true positive rate. For both misuse cases the false negative rate of our approach is acceptable. Only two out of 13 use cases for MUC2 were overlooked. Thus, we can state that our approach is sufficient to identify vulnerable use cases reliably. Although the results are good for the case study, further studies are needed in order to generalize results.

As the results of the second iteration compared to the first iteration imply, knowledge refinement can reduce the number of false positives sufficiently. In order to answer **RQ3**, we presented an extraction method guiding the requirements engineer to refine terms and properties and add new ones to the security knowledge. Therefore, it leverages heuristic findings obtained by our assessment approach. Although it requires some effort to handle case by case at the beginning, the knowledge saturates after a few iterations and the effort to refine knowledge decreases. Regarding long-living software system, we consider this a valuable investment. According to the results of the iTrust case study, our method is suitable to refine security knowledge which is organized according to **RQ1**. This supports reusing security knowledge gained during the development of security-critical software and feeding it back into requirements engineering.

Although the results indicate that our approach works, we have identified some limitations while conducting the iTrust case study. Regarding the security abstraction model, it is not feasible to encode attributes which are described in the use case implicitly. As an example, the description "The user enters the password." contains the trust level *user* and the asset *password*, but the entry point is implicitly encoded in the verb *enter*. If we know that the regarded system is a web application, we are able to imply that the entry point may be a *website*. Computer-based reasoning approaches, however, need this knowledge modeled beforehand. Moreover, the proposed knowledge structure cannot model simultaneous actions. For example: "While entering the required password, only the last

character is shown in clear text.” In some cases an incident may affect more than one use case. To identify these incidents, the interrelationship between use cases must be considered.

Nevertheless, our assessment approach gains appropriate results with reasonable computation costs. They can be considered as an initial indication of the presence of security loopholes followed by an in-depth security analysis.

VIII. RELATED WORK

AlHogail and Berri [2] propose the development of an architecture sustaining security knowledge within an organization. The architecture aims to manage tailored security processes, policies, and solutions. The goal is to capture and share security knowledge in order to efficiently react on security incidents and decrease dependencies on security experts. To capture security incidents as well as countermeasures, a pre-defined report template is used. Reports are analyzed manually to establish rules that are stored in an organization-wide knowledge base. Tsoumas and Gritzalis [41] suggest a security management approach for information systems which builds upon security knowledge gathered from various information sources. Raskin et al. [33] present an ontology-driven security approach. Ontologies are used to organize security knowledge (e.g. attacks and countermeasures, etc.) retrieved from natural language sources. Other approaches dealing with similar ontologies to manage security knowledge properly are presented in [23], [6]. In these approaches the ontology is mainly set up beforehand considering widely-accepted standards and information about the infrastructure of the information system as well as established policy documents.

Nuseibeh et al. [31] described their experiences applying a security requirements analysis. For this purpose, they developed a framework and associated tools that help to analyze the context systematically in which the system operates. It therefore offers different forms of structured argumentation. In Haley et al. [14], an approach to support security requirements elicitation and analysis is proposed. The method is based on constructing a system context using a problem-oriented notation. The system context is validated against the security requirements that are modeled as formal constraints. The approach reveals whether security requirements cannot be satisfied in the context or that the context is not sufficiently defined. Although the approach is very powerful, it requires a certain level of expertise to construct the context and understand the analysis results. To identify security requirements for certification and accreditation, Lee et al. [25] developed an approach based on extraction of relevant concepts from regulatory documents which are used to build up a problem domain ontology. Elahi et al. [9] proposed a vulnerability-centric requirements engineering framework for assessing the risk of vulnerabilities exploitation. Thus, this approach enables security requirements elicitation and analysis based on vulnerabilities. For this purpose, an extension of the i* meta-model to incorporate security-relevant concepts (e.g. attacks, countermeasures, vulnerabilities) is proposed. Moreover, a modeling process is proposed to attain the analysis model.

Sindre and Opdahl [39] have presented an approach to identify possible threats and analyze risks by eliciting misuse cases based on pre-defined templates. Another well-structured analysis method is to conduct security assessment by means of attack trees. An attack tree represents known attacks and their countermeasures in a tree structure. In Byres et al. [7], attack trees are used for assessing vulnerabilities in SCADA systems successfully. Jung et al. [21] proposed a case-based reasoning approach to incorporate past security accidents in risk analysis of e-commerce system. To apply this approach, the problem must be formulated in the well-structured case representation which requires additional effort.

In most approaches, evolution of security knowledge is not considered sufficiently. We are not aware of development approaches dealing with security knowledge and its evolution in a systematic way. To support requirements engineers by incorporating security knowledge semiautomatically, has not been considered in most approaches. Moreover, evolution of the regarded software system as well as knowledge changes has been insufficiently considered yet.

IX. CONCLUSIONS AND FUTURE RESEARCH

Maintaining information systems with many natural language requirements is a laborious task. Security related parts of a software have to be changed when requirements change or when assumptions about the system context do not hold any longer due to occurred security incidents.

In this paper, we presented a security assessment approach for supporting maintenance of long-living information systems independent of the process model, domain, or technology in use. According to our research questions, our approach is suitable to identify vulnerabilities in natural language requirements based on reported security incidents. It therefore provides support for avoiding repetition of security flaws which already occurred before. Moreover, we presented a method to refine or adapt security knowledge and to make it more suitable for future security assessments. In the iTrust case study, we have shown that our approach performs better than other approaches (Naive Bayes, SVM, k-NN). Although we only focus on technical aspects of security within this paper, our approach is capable of incorporating human-related aspects as well.

The success of our assessment approach highly depends on the quality of the security knowledge. Whereas detailed knowledge leads to more helpful information, it is expensive to retrieve and to model. To use our approach in an industrial setting, we have to evaluate a manageable level of detail that is used to document incidents. Moreover, we need to investigate whether intermediate feedback about security issues in requirements improves security of the software system. For this reason, we plan to conduct an industrial case study.

The presented approach is a foundation for the other parts of SecVolution. Identifying security issues in requirements and refining security knowledge in the requirements engineering process is important for our security aware evolution approach of long-living software systems.

ACKNOWLEDGMENT

This research is funded by the DFG project SecVolution (JU 2734/2-1 and SCHN 1072/4-1) which is part of the priority program SPP 1593 “Design For Future - Managed Software Evolution”.

REFERENCES

- [1] I. Alexander. Misuse cases: Use cases with hostile intent. *IEEE Software*, 20(1):58–66, 2003.
- [2] A. AlHogail and J. Berri. Enhancing it security in organizations through knowledge management. In *Proc. of the Int. Conference on Information Technology and e-Services (ICITeS)*, pages 1–6. IEEE, 2012.
- [3] G. Alvarez and S. Petrovic. A new taxonomy of web attacks suitable for efficient encoding. *Computers & Security*, 22(5):435–449, 2003.
- [4] V. Ambriola and V. Gervasi. Processing natural language requirements. *Automated Software Engineering*, pages 36–45, 1997.
- [5] S. Barnum and A. Sethi. Attack Patterns as a Knowledge Resource for Building Secure Software. In *OMG Software Assurance Workshop: Digital*, 2007.
- [6] P. Belsis, S. Kokolakis, and E. Kiountouzis. Information systems security from a knowledge management perspective. *Information Management & Computer Security*, 13(3):189–202, 2005.
- [7] E. Byres, M. Franz, and D. Miller. The use of attack trees in assessing vulnerabilities in SCADA systems. In *Proc. of the Int. Infrastructure Survivability Workshop*, 2004.
- [8] J. Eichler. Lightweight modeling and analysis of security concepts. In *Proc. of the Third Int. Conference on Engineering Secure Software and Systems*, ESSoS, pages 128–141. Springer, 2011.
- [9] G. Elahi, E. Yu, and N. Zannone. A vulnerability-centric requirements engineering framework: analyzing security attacks, countermeasures, and requirements based on vulnerabilities. *Requirements Engineering*, 15(1):41–62, 2009.
- [10] G. Elahi, E. Yu, and N. Zannone. A modeling ontology for integrating vulnerabilities into security requirements conceptual foundations. In *Proc. of the 28th Int. Conference on Conceptual Modeling*, pages 99–114, 2009.
- [11] C. Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.
- [12] S. Fenz and A. Ekelhart. Formalizing information security knowledge. In *Proc. of the 4th Int. Symposium on Information, Computer, and Communications Security (ASIACCS)*, page 183. ACM, 2009.
- [13] M. Guo and J. A. Wang. An ontology-based approach to model common vulnerabilities and exposures in information security. In *ASEE Southeast Section Conference*, 2009.
- [14] C. B. Haley, R. C. Laney, J. D. Moffett, and B. Nuseibeh. Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Trans. Software Eng.*, 34(1):133–153, 2008.
- [15] S. Hansman and R. Hunt. A taxonomy of network and computer attacks. *Computers & Security*, 24(1):31–43, 2005.
- [16] A. Herzog, N. Shahmehri, and C. Duma. An ontology of information security. *IJISP*, 1(4):1–23, 2007.
- [17] S. H. Houmb, S. Islam, E. Knauss, J. Jürjens, and K. Schneider. Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec. *Requirements Engineering*, 15(1):63–93, 2009.
- [18] J. D. Howard and T. A. Longstaff. A common language for computer security incidents. Technical report, Sandia National Laboratories, 1998.
- [19] A. Jain and D. Shanbhag. Addressing security and privacy risks in mobile applications. *IT Professional*, (October):28–33, 2012.
- [20] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of International Conference Research on Computational Linguistics (ROCLING)*, 1997.
- [21] C. Jung, I. Han, and B. Suh. Risk analysis for electronic commerce using case-based reasoning. *Int. Journal of Intelligent Systems in Accounting, Finance & Management*, 8(1):61–73, Mar. 1999.
- [22] M. Karyda, T. Balopoulos, L. Gymnopoulos, S. Kokolakis, C. Lambrinoudakis, S. Gritzalis, and S. Dritsas. An ontology for secure e-government applications. In *Proc. of the First Int. Conference on Availability, Reliability and Security*, pages 1033–1037. IEEE, 2006.
- [23] S. Kesh and P. Ratnasingam. A knowledge architecture for IT security. *Communications of the ACM*, 50(7), 2007.
- [24] E. Knauss, D. Lübke, and S. Meyer. Feedback-Driven Requirements Engineering: The Heuristic Requirements Assistant. In *Proc. of the 31st Int. Conference on Software Engineering (ICSE)*, pages 587–590, 2009.
- [25] S.-W. Lee, R. Gandhi, D. Muthurajan, D. Yavagal, and G.-J. Ahn. Building problem domain ontology from security requirements in regulatory documents. In *Proc. of the Int. workshop on Software engineering for secure systems (SESS)*, pages 43–50. ACM, 2006.
- [26] H. Mcmanus and P. D. Hastings. A Framework for Understanding Uncertainty and its Mitigation and Exploitation in Complex Systems. In *15th Annual In. Symposium of the Int. Council on Systems Engineering (INCOSE)*, pages 1–20, 2005.
- [27] A. Meneely, B. Smith, and L. Williams. iTrust electronic health care system case study. In *Software and Systems Traceability*, pages 425–438. Springer, 2012.
- [28] A. Miede, N. Nedyalkov, C. Gottron, A. König, N. Repp, and R. Steinmetz. A Generic Metamodel for IT Security Attack Modeling for Distributed Systems. *Int. Conference on Availability, Reliability and Security (ARES)*, pages 430–437, 2010.
- [29] H. Mouratidis, P. Giorgini, and G. Manson. An ontology for modelling security: The tropos approach. In V. Palade, R. Howlett, and L. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 2773 of *LNCS*, pages 1387–1394. Springer, 2003.
- [30] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48:443–453, 1970.
- [31] B. Nuseibeh, C. B. Haley, and C. Foster. Securing the Skies: In Requirements We Trust. *IEEE Computer*, 42(9):64–72, 2009.
- [32] T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet: Similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL*, pages 38–41. Association for Computational Linguistics, 2004.
- [33] V. Raskin, C. F. Hempelmann, K. E. Trienzenberg, and S. Nirenburg. Ontology in information security: a useful theoretical foundation and methodological tool. In *Proc. of the Workshop on New security paradigms*, NSPW, pages 53–59. ACM, 2001.
- [34] T. Ruhroth, S. Gärtner, J. Bürger, J. Jürjens, and K. Schneider. Versioning and evolution requirements for model-based system development. In *Int. Workshop on Comparison and Versioning of Software Models (CVSM)*, 2014.
- [35] H. Schmid. Probabilistic Part-of-Speech tagging using decision trees. In *Proc. of the Int. Conference on New Methods in Language Processing*, pages 44–49, 1994.
- [36] K. Schneider. *Experience and Knowledge Management in Software Engineering*. Springer, 2009.
- [37] K. Schneider, E. Knauss, S. Houmb, S. Islam, and J. Jürjens. Enhancing security requirements engineering by organizational learning. *Requirements Engineering*, 17(1):35–56, 2011.
- [38] S. Shiva, C. Simmons, C. Ellis, D. Dasgupta, S. Roy, and Wu. AVOIDIT: A cyber attack taxonomy. Technical report, University of Memphis, August, 2009.
- [39] G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, 2005.
- [40] F. Swiderski and W. Snyder. *Threat Modeling*. Microsoft Press Corp., 2004.
- [41] B. Tsoumas and D. Gritzalis. Towards an Ontology-based Security Management. In *Proc. of the 20th Int. Conference on Advanced Information Networking and Applications (AINA)*, volume 1, pages 985–992. IEEE, 2006.
- [42] J. Undercoffer, A. Joshi, and J. Pinkston. Modeling computer attacks: An ontology for intrusion detection. In *6th Int. Symposium on Recent Advances in Intrusion Detection*, pages 113–135. Springer, 2003.
- [43] K. H. Vellani. Vulnerability Assessments. In *Strategic Security Management: A Risk Assessment Guide for Decision Makers*, pages 85–107. 2007.
- [44] WORDNET Webpage. Glossary of terms used in WordNet system , 2014.

Rationalism with a Dose of Empiricism: Case-Based Reasoning for Requirements-Driven Self-Adaptation

Wenyi Qian^{*†}, Xin Peng^{*†}, Bihuan Chen^{*†}, John Mylopoulos[‡], Huanhuan Wang^{*†}, and Wenyun Zhao^{*†}

^{*}School of Computer Science, Fudan University, Shanghai, China

[†]Shanghai Key Laboratory of Data Science, Fudan University, China

[‡]Department of Information Engineering and Computer Science, University of Trento, Italy

{qianwy, pengxin, bhchen, huanhuanwang13, wyzhao}@fudan.edu.cn, jm@disi.unitn.it

Abstract—Requirements-driven approaches provide an effective mechanism for self-adaptive systems by reasoning over their runtime requirements models to make adaptation decisions. However, such approaches usually assume that the relations among alternative behaviours, environmental parameters and requirements are clearly understood, which is often simply not true. Moreover, they do not consider the influence of the current behaviour of an executing system on adaptation decisions. In this paper, we propose an improved requirements-driven self-adaptation approach that combines goal reasoning and case-based reasoning. In the approach, past experiences of successful adaptations are retained as adaptation cases, which are described by not only requirements violations and contexts, but also currently deployed behaviours. The approach does not depend on a set of original adaptation cases, but employs goal reasoning to provide adaptation solutions when no similar cases are available. And case-based reasoning is used to provide more precise adaptation decisions that better reflect the complex relations among requirements violations, contexts, and current behaviours by utilizing past experiences. Our experimental study with an online shopping benchmark shows that our approach outperforms both requirements-driven approach and case-based reasoning approach in terms of adaptation effectiveness and overall quality of the system.

I. INTRODUCTION

A self-adaptive system can switch among alternative behaviours, so that it can continue to satisfy stakeholder requirements (goals) in a changing and uncertain runtime environment. Requirements-driven approaches [1], [2], [3], [4], [5] provide an effective mechanism for such self-adaptive systems by reasoning over their runtime requirements models to make adaptation decisions.

However, such approaches usually assume that the requirements of a self-adaptive system are well understood so that they can be used as the basis for an adaptation mechanism. In particular, the relations among alternative behaviours, environmental parameters and requirements are clearly understood and modelled. Given these models, the adaptation mechanism can reason about alternative behaviours that can work best when requirements or contexts (environmental parameters) change at runtime.

The above assumption, however, is often simply not true when the system is complex and the relations among behaviours, contexts and requirements are not precisely specified. Moreover, an appropriate adaptation decision depends not only on requirements and context changes but also on the current

behaviour deployed in the system. In that sense, there can be very complex combinations of requirements violations, contexts, and currently deployed behaviours (current configuration of the system), each of which may demand a different adaptation solution.

By contrast, case-based reasoning (CBR) is able to utilize specific knowledge of previously experienced, concrete problem situations (cases) instead of relying solely on general knowledge [6]. Its underlying idea is that new problems can be solved by reusing solutions of the past to similar problems. This makes CBR a promising problem solving paradigm when the problem is not very well understood or the knowledge is hard to express. Several attempts have been made to apply CBR in self-adaptive systems [7], [8], [9], where cases are described by low-level symptoms (symbolic or numeric variables) characterizing encountered problems (e.g. requirements failures). And they depend on a set of original cases that are manually constructed by humans.

In this paper, we propose an improved requirements-driven self-adaptation approach that combines goal reasoning and case-based reasoning. In the approach, past experiences of successful adaptations are retained as adaptation cases and stored in a case base. Adaptation cases are described by not only requirements violations (a set of quality attributes) and contexts (a set of environmental parameters), but also currently deployed behaviours. Based on a predefined requirements goal model, both the current behaviour and the adaptation solution (i.e. new behaviour to be adopted after adaptation) in a case are specified by goal configurations consisting of specific alternatives for variation points and values for control variables. Such a goal-oriented case representation provides richer and higher-level representation for adaptation cases. Instead of depending on a set of original adaptation cases, our approach employs goal reasoning to provide adaptation solutions when no similar cases are available. Case-based reasoning, on the other hand, provides more precise adaptation decisions that better reflect the complex relations among requirements violations, contexts, and current behaviours by utilizing past experiences (adaptation cases).

To evaluate the effectiveness of our approach, we conduct an experimental study with an online shopping benchmark. The results show that our approach outperforms both a requirements-driven approach and a case-based reasoning

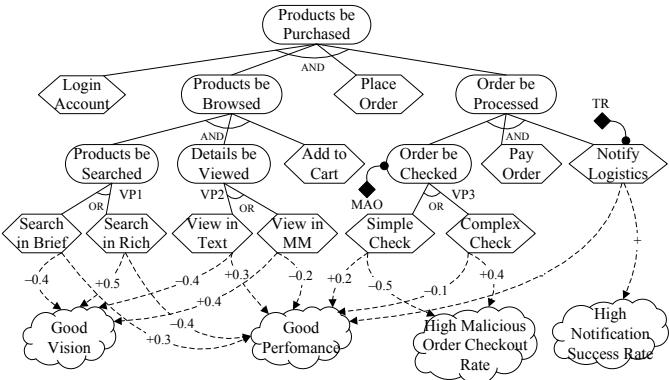


Fig. 1. The Parameterized Goal Model of an Online Shopping System

approach in terms of adaptation effectiveness and overall quality of the system.

The rest of the paper is structured as follows. Section II introduces the required preliminaries including parameterized goal models and case-based reasoning. Section III presents the proposed self-adaptation approach. Section IV evaluates the proposed approach using an online shopping system and discusses related issues. Section V reviews a number of existing proposals and compares them with ours. Finally, Section VI concludes the paper and outlines our future work.

II. PRELIMINARIES

In this section, we briefly introduce the required preliminaries, i.e. parameterized goal models and case-based reasoning.

A. Parameterized Goal Models

In goal-oriented requirements analysis, functional requirements are modeled as *hard goals*, and quality requirements are modeled as *softgoals* [10]. In a requirements goal model, goals are iteratively refined into subgoals through AND/OR *decomposition links* until we reach goals that are simple enough to be fulfilled by *tasks* carried out by software or human agents. To fulfill an AND/OR-decomposed goal, all/at least one of its subgoals must be satisfied. Furthermore, goals can be related to each other through quantitative *contribution links* $w+$ and $w-$ where the weight w is between 0 and 1 [11]. $+/-$ indicates that the satisfaction of the source goal contributes to w -level satisfaction/denial of the target goal.

In a goal model, OR-decomposed goals represent a kind of variation points of system behaviours with their alternative subgoals as the variants, and thus provide a mechanism for reconfiguration. Recently, control variables were introduced as attachments to goals or tasks to represent variables that influence the fulfilment of a goal, or the execution of a task [12]. Control variables provide another mechanism for reconfiguration. Goal variation points and control variables are together referred to parameters, and such goal models are called parameterized goal models. Here, a goal configuration that satisfies a root-level goal consists of choosing one of the alternatives for every variation point, and also choosing a value for every control variable. The alternative behaviours

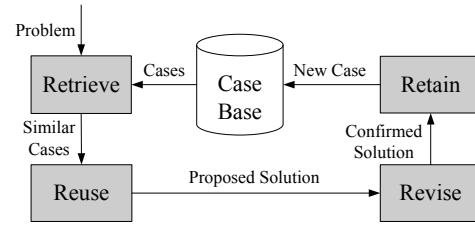


Fig. 2. The Process of Case-Based Reasoning

of a system at runtime can then be represented by alternative goal configurations.

Figure 1 shows the parameterized goal model of an online shopping system with three variation points and two control variables. For example, the variation point *Products be Searched* can be satisfied by either *Brief Search* (simply search with keywords) or *Rich Search* (search with recommendation), which have reverse contributions to *Good Usability* and *Good Performance*. The control variable *MAO* represents the minimum amount for an order that needs to be checked before further processing, while the control variable *NR* represents the number of retries to be attempted if a notification to logistics fails. Then a candidate goal configuration that satisfies *Products be Purchased* is [*Rich Search*, *View in MM*, *Complex Check*, 200 (*MAO*), 2 (*NR*)].

B. Case-Based Reasoning

Case-based reasoning (CBR) is a problem solving paradigm, which is able to use the knowledge of previously experienced, concrete problem situations (cases) to solve new problems [6]. CBR is well suited for problems where requirements and domain knowledge is simply unavailable. Further, CBR is actually an incremental and sustained learning process that retains and reuses knowledge from past experiences [6].

Figure 2 shows the CBR process, which comprises four steps, i.e. *Retrieve*, *Reuse*, *Revise* and *Retain*. A case base is used to store previously learned cases in the form of problem-solution pairs.

In particular, *Retrieve* retrieves cases whose problems are similar to the problem at hand, using a matching operation. Based on the retrieved similar cases, *Reuse* is conducted to produce a solution to the given problem, for example, by calculating averages of the solutions of similar cases. Then in the *Revise* step, the solution to the given problem is tested for success, e.g. by being applied to real-world problems or evaluated by experts. If failed, the solution may also be repaired using domain-specific knowledge such as rules. Finally, *Retain* retains the learned new case by incorporating it into the case base for future reuse.

III. OUR APPROACH

This section first presents an overview of the proposed approach, and then details the underlying techniques.

A. Overview

Our approach combines requirements-driven self-adaptation and case-based reasoning. An adaptation case base is used

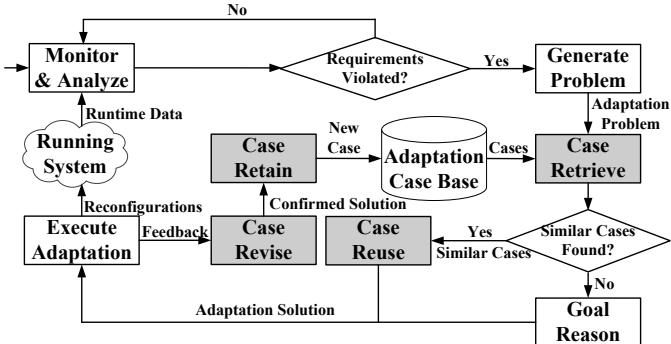


Fig. 3. Approach Overview

to store past experiences of successful adaptations, which is initially empty. On the other hand, a requirements goal model is used to provide a richer and higher-level representation for adaptation cases and reason about adaptation solutions when no similar past cases can be found. Figure 3 presents an overview of our approach, showing the main steps and input/output in the process. The four gray rectangles in the figure correspond to the four main tasks of case-based reasoning, i.e. *Retrieve*, *Reuse*, *Revise*, *Retain*.

Based on a running system, our approach creates an adaptation control loop that is periodically executed. In each adaptation process, quality attributes (e.g. response time, success rate, cost) and environmental parameters (e.g. access load, CPU/memory usage) of the system are monitored and analyzed based on collected runtime data. Then the approach evaluates the quality attributes and determines whether some requirements are violated, for example, quality constraints are violated or desired quality decreases. If requirements violations are detected, an adaptation problem is raised, which describes the detected quality attributes, contexts, and the current behaviour adopted by the system.

To find an adaptation solution for an adaptation problem, the approach retrieves similar adaptation cases from the case base. If similar cases are found, an adaptation solution (i.e. goal configuration) is synthesized from them (case reuse) for the current adaptation problem. If not found, a goal reasoner is used to produce an adaptation solution by reasoning about optimal goal configurations based on the predefined goal model. Based on the adaptation solution produced by case reuse or goal reasoning, the approach adapts the running system by reconfiguring the runtime architecture and parameters.

After an adaptation solution is executed, the changes of related quality attributes of the system are collected as feedback. Based on the feedback, the effect of the adaptation is evaluated by the case revision mechanism. If the adaptation is successful, i.e. desired quality attributes are improved, the adaptation solution used is confirmed and sent to the case retention mechanism. Then a new case is created for the current adaptation and retained in the case base.

B. Case Representation

Each case in our approach represents a successful adaptation in which a proper switch of alternative system behaviours

is made under a specific situation (requirements violations, contexts, and currently deployed behaviour). In case-based reasoning, a case usually includes three major parts [13]: the *problem-situation description*, the state of the world and the problem needing to be solving; the *solution*, the stated or derived solution to the problem; the *effect*, the resulting state of the world when the solution was carried out. In our approach, the solution part of an adaptation case specifies a switch to another alternative goal configuration. And the effect part of an adaptation describes the outcome of an adaptation case according to the improvement of quality attributes. Therefore, an adaptation case can be characterized by the following representation:

$$\text{Case} = \langle \text{Problem}, \text{GoalConfig}_{\text{new}}, \text{Quality}_{\text{new}} \rangle \quad (1)$$

1) *Adaptation Problem*: As specified in the following equation, the problem description (*Problem*) of an adaptation case consists of the situation of requirements violations (*Quality_{cur}*) and contexts (*Context_{cur}*), and an additional description about the goal configuration currently adopted (*GoalConfig_{cur}*) when an adaptation problem is raised.

$$\text{Problem} = \langle \text{Quality}_{\text{cur}}, \text{Context}_{\text{cur}}, \text{GoalConfig}_{\text{cur}} \rangle \quad (2)$$

Among the problem description, *Quality_{cur}* denotes the situation of requirements violations specified by a set of quality attributes, each corresponding to a softgoal. And for each quality attribute a value is aggregated based on the runtime data collected during a fixed adaptation period (e.g. 3 minutes). For example, for the softgoal *Good Performance* in Figure 1, a value of average response time can be aggregated and assigned to it. *Context_{cur}* denotes the situation of contexts specified by a set of predefined environmental parameters. For example, the runtime contexts of a Web system usually can be described by CPU usage, memory usage, access load etc. *GoalConfig_{cur}* denotes the current goal configuration that is adopted for the root goal of the system when an adaptation problem is raised. It is specified by a set of alternative goals/tasks and control variable values according to a predefined goal model. For example, according to the goal model in Figure 1, a description of *GoalConfig_{cur}* consists of an alternative task chosen for *Products be Searched*, *Details be Viewed*, and *Order be Checked* respectively, and a value for *MAO* (minimum amount of order) and *NR* (number of retries) respectively.

2) *Adaptation Solution*: The solution part (*GoalConfig_{new}*) of an adaptation case specifies a switch to another alternative goal configuration of the system. Therefore, it is described in the same way of a current goal configuration (*GoalConfig_{cur}*), i.e. by a set of alternative goals/tasks and control variable values.

3) *Adaptation Effect*: The effect description (*Quality_{new}*) of an adaptation case records the improvement of quality attributes after the adaptation is carried out. Therefore, it is described in the same way of the quality part (*Quality_{cur}*) of an adaptation problem, i.e. by the values of a set of quality attributes.

C. Goal Reasoning

Goal reasoning is used to provide adaptation solutions based on a predefined goal model when no similar cases are available. To reflect the problem of requirements violations, the weights of related quality attributes (softgoals) are dynamically tuned to improve violated requirements. The procedure of weight tuning is described in Algorithm 1. The input of the algorithm is a set of quality attributes $QASet$. Each quality attribute is described using the following representation, where $value, cons_u, cons_l, weight$ represent the current value, upper constraint (the value of the quality attribute should be lower than $cons_u$), lower constraint (the value of the quality attribute should be higher than $cons_l$), and current weight of the quality attribute. The upper (lower) constraint is assigned to positive (negative) infinity if the quality attribute has no upper (lower) constraint. The weights of all the quality attributes are initiated to 1 when the system is launched.

$$\text{QualityAttribute} = \langle value, cons_u, cons_l, weight \rangle \quad (3)$$

The algorithm identifies quality attributes whose constraints are violated ($VioSet$). If some quality constraints are violated (i.e. $VioSet$ is not empty), the weights of related quality attributes are increased by a predefined constant α (e.g., 20%). Otherwise, it means that the current adaptation is triggered by the degradation of the combined utility, so the weight of a selected quality attribute with the lowest weight is increased by α . Note that the weight of each quality attribute (corresponding to a softgoal) will be normalized into a relative weight to be used in goal reasoning.

Algorithm 1 Weight Tuning

```

1: procedure WEIGHTTUNING( $QASet$ )
2:    $VioSet = \emptyset, min = +\infty$ 
3:   for each  $q \in QASet$  do
4:     if  $q.value > q.cons_u \text{ || } q.value < q.cons_l$  then
5:        $VioSet = VioSet \cup \{q\}$ 
6:     else if  $q.weight < min$  then
7:        $min = q.weight$ 
8:        $temp = q$ 
9:     end if
10:   end for
11:   if  $VioSet \neq \emptyset$  then
12:     for each  $q \in VioSet$  do
13:        $q.weight = q.weight \times (1 + \alpha)$ 
14:     end for
15:   else
16:      $temp.weight = temp.weight \times (1 + \alpha)$ 
17:   end if
18: end procedure

```

To involve control variables in goal reasoning, we transform each task with a control variable into an OR-decomposition as shown in Figure 4. For each task $Task$ with a control variable CV , three alternative tasks $Task(CV++)$, $Task(CV-)$, $Task(CV0)$ are created, representing the same $Task$ but with

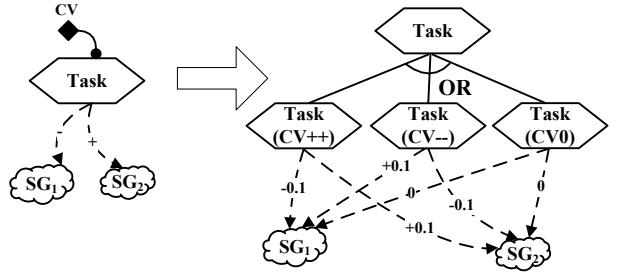


Fig. 4. Transform Control Variable into OR-Decomposition

increasing, decreasing, and unchanged CV respectively. For each of them, a contribution link is created with related softgoals. The contribution weights of $Task(CV0)$ to a softgoal are always 0, indicating that there is no influence change on the softgoal if CV remains unchanged. If $Task(CV++/-)$ is selected, CV will be increased/decreased with a predefined changing pace. And if the value of CV goes beyond its value range after increase/decrease, the current goal configuration under consideration is discarded.

For a candidate goal configuration, the satisfaction level of a softgoal can be calculated using label propagation algorithms [11]. And the overall contribution of the goal configuration to all the softgoals is calculated as the weighted sum of the satisfaction levels of all the softgoals. The weight of a softgoal is the relative weight of its corresponding quality attribute calculated as the following equation.

$$\text{RelativeWeight}(qa) = \frac{qa.weight}{\sum_{q \in QASet} q.weight}$$

Based on the contribution calculation, the goal configuration with the highest overall contribution is chosen as the result of goal reasoning and used as the solution for the current adaptation problem.

D. Case Retrieval

Case retrieval is based on the distance between a given adaptation problem and the problem description of each case in the adaptation case base. The distance of two adaptation problem P_1 and P_2 is calculated as the Euclidean distance of their n -dimension description vectors V_{P_1} and V_{P_2} :

$$\text{Distance}(P_1, P_2) = \sqrt{\sum_{i=1}^n (\text{diff}(V_{P_1}[i], V_{P_2}[i]))^2} \quad (4)$$

The diff function in the above equation denotes the difference of the values on the i th dimension in P_1 and P_2 . For a dimension corresponding to a enumeration type (e.g. alternative goal/task), the difference is 0 if the two values are the same and 1 if not. For a dimension corresponding to a number type (e.g. control variable), the difference is computed after normalization. The normalization of a numerical variable can be calculated by $\frac{v - MIN}{MAX - MIN}$, where v is its current value, MIN and MAX are its minimal and maximal values during the system operation.

Based on the case distance measurement, the case retrieval procedure (see Algorithm 2) identifies similar adaptation cases with a predefined higher distance threshold $threshold_h$ and a predefined lower threshold $threshold_l$ ($threshold_l < threshold_h$). The inputs of the algorithm are an adaptation problem $curProblem$ and a set of adaptation cases $CaseSet$. It returns a perfect adaptation case or a set of similar-enough cases $SimCases$. If there are some cases whose problem descriptions are very similar (distance lower than $threshold_l$) to the given problem, the most similar case of them is returned as the result of case retrieval. Otherwise, a set of similar-enough cases $SimCases$ (distance lower than $threshold_h$) is returned as the result.

Algorithm 2 Adaptation Case Retrieval

```

1: function RETRIEVE( $curProblem$ ,  $CaseSet$ ,  $SimCases$ )
2:    $min = threshold_l$ ,  $perfectCase = null$ 
3:   for each  $case \in CaseSet$  do
4:      $dis = Distance(curProblem, case.problem)$ 
5:     if  $dis < min$  then
6:        $perfectCase = case$ 
7:        $min = dis$ 
8:     else if  $dis < threshold_h$  then
9:        $SimCases = SimCases \cup \{case\}$ 
10:    end if
11:   end for
12:   return  $perfectCase$ 
13: end function

```

E. Case Reuse

The case reuse strategy of our approach depends on the result of case retrieval. If a perfect case is returned, the solution part of the returned case (i.e. a goal configuration) is directly adopted as the adaptation solution. Otherwise, our approach synthesizes an adaptation solution based on the returned similar cases.

To avoid the chaos brought by mixing up too many cases, our approach only picks up to two most similar cases among returned cases. Besides, another goal configuration is generated using goal reasoning. The goal configurations of the two selected cases together with the generated goal configuration are synthesized into an adaptation solution. The synthesis is done for each OR-decomposed goal and each control variable involved in the goal configurations. An OR-decomposed goal is synthesized by voting, i.e. alternative goals/tasks adopted by the most goal configurations are selected. If more than one alternative goals/tasks are selected, an alternative is randomly selected as the result. A control variable is synthesized by calculating the average of its values in different goal configurations.

F. Case Revision and Retention

For an adaptation solution with the goal configuration $goalConfig$ for an adaptation problem $problem$, the case revision mechanism evaluates its effectiveness. Based on the

feedback from adaptation execution, the evaluation measures related quality attributes of the system during the adaptation period after the adaptation execution. The obtained situation of quality attributes $Quality_{new}$ is then compared with the situation in the adaptation problem (i.e. $problem.Quality_{cur}$). If $problem$ involves violations of quality constraints, the effectiveness of the adaptation solution is judged by the following criterion: no new constraint violations are involved in $Quality_{new}$ and all of the violated quality attributes in $problem.Quality_{cur}$ have been improved. Otherwise, the effectiveness is judged by the improvement of the combined utility.

If the adaptation solution is evaluated to be effective, a new adaptation case is created with $problem$ as the problem, $goalConfig$ as the solution, and $Quality_{new}$ as the effect. If no past cases with the same problem description exist in the case base, the new case is retained for future reuse, i.e. added to the case base. If there is a past case with the same problem description and the effect of the new case is better than that of the old one, the old case is replaced by the new case. Otherwise, the new case is discarded.

IV. EXPERIMENTAL STUDY

To evaluate the effectiveness of the proposed approach, we conduct an experimental study based on an online shopping system, whose goal model is shown in Figure 1, to answer the following two research questions:

- *Q1*: Can the proposed approach achieve improvements over self-adaptation approaches that involve only goal reasoning or case-based reasoning? (Comparative Evaluation)
- *Q2*: Can goal reasoning and case-based reasoning be effectively combined in the adaptation process of the proposed approach? (Process Analysis)

For Q1, we will compare the proposed approach with other two approaches involving only goal reasoning or case-based reasoning, to evaluate if the proposed approach could perform better. The comparison will be conducted in two ways: the effectiveness of the adaptation solutions and the improvement of response-time and utility of the system. For Q2, we will analyze how the proposed approach carries out during the self-adaptation process, to check if the combination of goal reasoning and case-based reasoning could really work well.

A. Experimental Setup

In the experiments, twelve environmental parameters and four quality attributes were monitored to facilitate runtime adaptations. Table I presents the environmental parameters considered in our experiments. These parameters were obtained by Linux's virtual memory statistics. The quality attributes for the softgoals of the online shopping system and their corresponding metrics are detailed as follows.

- The metric for the softgoal *Good Performance* is response time, which was measured by log analysis as the average response time of each request in an interval. For example, there are five requests with the response time being 100

TABLE I
ENVIRONMENTAL PARAMETERS OF THE ONLINE SHOPPING SYSTEM

| Context Variables | Description |
|-------------------|--|
| thread_num | the number of concurrent access threads |
| c_procs_r | the number of processes waiting for run time |
| c_procs_b | the number of processes in uninterruptible sleep |
| c_memory_swpd | the amount of virtual memory used |
| c_memory_free | the amount of idle memory |
| c_memory_buff | the amount of memory used as buffers |
| c_memory_cache | the amount of memory used as cache |
| c_io_bi | blocks received from a block device (blocks/second) |
| c_io_bo | blocks sent to a block device (blocks/second) |
| c_system_in | the number of interrupts per second, including the clock |
| c_system_cs | the number of context switches per second |
| c_cpu_id | time spent idle of total CPU time |

ms, 200 ms, 150 ms, 300 ms and 250 ms, then the average response time is 200 ms.

- The metric for the softgoal *Good Usability* is user satisfaction, which was measured by a simulated real-life customer feedback analysis. Here we assumed that user satisfaction for a searching request was a random value between 0.0 and 0.5 if the variation point *VP1* in Figure 1 was bound to *Brief Search*, and between 0.5 and 1.0 if the *VP1* was bound to *Rich Search*. Similarly, we assumed that user satisfaction for a viewing request was a random value between 0.0 and 0.5 if the variation point *VP2* was bound to *View in Text*, and between 0.5 and 1.0 if the *VP2* was bound to *View in MM*. And the average of all the user satisfaction of both searching and viewing request will be the final quality metric. For example, if *VP1* and *VP2* are bound to *Brief Search* and *View in Text* respectively, and there are two searching requests whose user satisfactions are 0.2 and 0.3 and two viewing requests whose user satisfactions are 0.1 and 0.2, then the user satisfaction is $\frac{0.2+0.3+0.1+0.2}{4} = 0.2$.
- The metric for the softgoal *High Malicious Order Checkout Rate* is the rate of identified malicious order in all malicious orders. Here we assumed that the alternative task *Simple Check* of *VP3* can check out 50% malicious orders, *Complex Check* can check out 80% malicious orders, and each order has a 10% chance to be a malicious order. Besides, the control variable *MAO* determines which orders will be checked. For example, there are 100 orders and 10 of them are malicious orders. Among these 10 malicious orders, 8 orders' amount is higher than 1000 dollars. Given that *VP3* is bound to *Simple Check* and the *MAO* is set to 1000 dollars, the malicious order rate will be $\frac{8*50\%}{10} = 0.4$.
- The metric for the softgoal *High Notification Success Rate* is the rate of successfully-notified orders in all orders. For example, there are 10 orders and 8 of them are successfully notified to logistics, then the notification success rate is 80%. Note that a notification can be successfully sent after several retries and the control variable *NR* determines the maximum numbers of retries.

For response time, we associated it with a quality constraint. We defined a combined utility of the other three quality attributes (i.e. user satisfaction, malicious order checkout rate, and notification success rate) as the weighted sum of their normalized values (between 0 and 1) for another quality constraint. And the weights of the three quality attributes were set to 0.4, 0.4, 0.2, respectively.

Measuring the values by monitoring quality attributes in each interval, the adaptation mechanism determines whether an adaptation is required for requirements violations based on the following two conditions:

- if the response time is larger than 1000 ms
- if the combined utility is smaller than 0.45 and the response time is smaller than 600 ms

Once one of the two conditions is met, the adaptation mechanism will determine and execute an appropriate reconfiguration to improve related quality attributes. We say an adaptation is *effective* if the violated requirements (the response time or utility) are improved in the following interval.

The behavior of the online shopping system has three variation points and two control variables (see Figure 1). In the experiments, we set the range of the control variables *MAO* and *NR* being 0 to 2000 and 0 to 10, with the changing pace being 200 and 1. The initial behavior of the system was set to [*Brief Search*, *View in Text*, *Simple Check*, 1000 (*MAO*), 5 (*NR*)].

To evaluate the effectiveness of the proposed approach, we conducted three experiments with the following three approaches respectively using the same experimental settings.

- *Goal-Based Approach*: The self-adaptation approach that uses goal reasoning to produce adaptation solutions.
- *Case-Based Approach*: The self-adaptation approach that uses case-based reasoning to produce adaptation solutions.
- *Proposed Approach*: The proposed self-adaptation approach that combines goal reasoning and case-based reasoning.

The original case base of the *Case-Based Approach* is constructed by goal reasoning, i.e. retaining adaptation solutions produced by goal reasoning as cases. And the case reuse strategy is to select and reuse the most similar case.

We conducted each of the experiments for 3.5 hours. The adaptation interval was set to one minute, i.e. the adaptation mechanism was periodically executed every one minute. In the experiment with the *Case-Based Approach*, the initial training phase for the construction of the original case base took about 55 minutes. In the experiment with the *Proposed Approach*, the two similarity thresholds $threshold_l$ and $threshold_h$ were set to 0.5 and 1.0 respectively. All the three experiments were conducted on the same server with a 4-core 3.1 GHz CPU and 8 GB RAM. We used JMeter to simulate concurrent system accesses and the same simulation script was applied to all the three experiments.

TABLE II
ADAPTATION FREQUENCY OF THE THREE APPROACHES

| Approach | Adaptation (#) | Eff. (#) | Rate (%) |
|---------------------|----------------|----------|----------|
| Goal-Based Approach | 62 | 33 | 53 |
| Case-Based Approach | 48 | 31 | 65 |
| Proposed Approach | 49 | 35 | 71 |

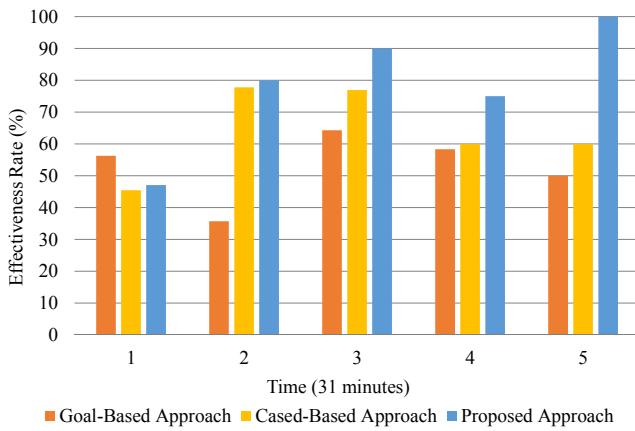


Fig. 5. Distribution of the Effectiveness Rate of the Three Approaches

B. Comparative Evaluation (Q1)

To fairly compare the effectiveness of the three approaches, we collected their frequency of adaptations which were triggered in the last 155 minutes of 3.5 hours (i.e. after the training phase of the *Case-Based Approach*). Table II shows the adaptation frequency of the three approaches. The first column lists the involved approaches, the second column lists the number of adaptations, the third column lists the number of effective adaptations, and the fourth column lists the effectiveness rate of the adaptations.

We can observe that the *Goal-Based Approach* triggered 62 adaptations, and 33 of them were evaluated to be effective; the *Case-Based Approach* triggered 48 adaptations, and 31 of them were evaluated to be effective; and the *Proposed Approach* triggered 49 adaptations, and 35 of them were evaluated to be effective. Moreover, the *Goal-Based Approach* achieved the lowest effectiveness rate (i.e. 53%), and the *Proposed Approach* achieved the highest effectiveness rate (i.e. 71%). This shows that it will be less effective if the adaptation mechanism only utilizes the rationalism knowledge underlying goal reasoning or the empiricism knowledge underlying case-based reasoning.

In addition, we divided the 155 minutes into five time slots, and analyzed the effectiveness rate of the adaptations of the three approaches during each time slot. Figure 5 shows the distribution of the effectiveness rate of the three approaches. The X axis denotes the time slot, and the Y axis denotes the effectiveness rate of the three approaches.

We can see that the *Case-Based Approach* mostly achieved a higher effectiveness rate than the *Goal-Based Approach* because it can provide more precise adaptation decisions, but it always achieved a lower effectiveness rate than the *Proposed Approach* because it lacks the rationalism support. In

addition, the *Proposed Approach* always achieved the highest effectiveness rate except for the first time slots. This is because the *Proposed Approach* did not have a rich enough case base in the beginning to handle new emerging problems. However, in the long run, the *Proposed Approach* is more effective than both the *Case-Based Approach* and the *Goal-Based Approach* because it combines the rationalism knowledge underlying goal reasoning and the empiricism knowledge underlying case-based reasoning.

Furthermore, Figure 6 and 7 respectively show the response time and utility of the three approaches. The X axis denotes the adaptation interval, and the Y axis denotes the response time and utility per adaptation interval. The triangle/square marks on these curves mean that an effective/ineffective adaptation was triggered at that adaptation interval. The vertical dashed lines represent the end of the training phase of the *Case-Based Approach*. The horizontal dashed lines represent the threshold of response time (i.e. 1000 ms) and utility (i.e. 0.45).

We can observe from the curves that the *Proposed Approach* triggered the adaptations effectively in most times, i.e. either to reduce the response time, or to increase the utility at the price of performance degradation. The *Goal-Based Approach* and *Case-Based Approach* were less effective. Numerically, the rate of satisfying the constraint of response time (i.e. less than 1000 ms) in the last 155 minutes was 89% (138/155) for the *Proposed Approach*, 90% (140/155) for the *Goal-Based Approach*, and 90% (140/155) for the *Case-Based Approach*. On the other hand, the rate of achieving a better utility (i.e. larger than 0.45) when the response time is less than 1000 ms was 51% (79/155) for the *Proposed Approach*, 43% (67/155) for the *Goal-Based Approach*, and 44% (68/155) for the *Case-Based Approach*. This shows that the proposed approach outperforms *Goal-Based Approach* and *Case-Based Approach* in utility while keeping almost the same violation rate of the constraint of response time.

The above observations from Table II and Figure 5, 6 and 7 answer Q1 positively that the proposed approach can improve the effectiveness of adaptations and the overall utility by combining the rationalism underlying goal reasoning and the empiricism underlying case-based reasoning.

C. Process Analysis (Q2)

To analyze the adaptation process of the proposed approach, we distinguish the reasoning mechanisms of adaptations. Specifically, adaptations can be generated by case-based reasoning (using the perfect case), or goal reasoning (using the best goal configuration), or synthesized reasoning (synthesizing the two most similar cases and the best goal configuration).

Table III shows the frequency of the adaptations of different reasoning mechanisms in the proposed approach in the whole 3.5 hours. The first column lists the reasoning mechanisms, the second and third list the number of adaptations and the number of effective adaptations, and the last column lists the effectiveness rate.

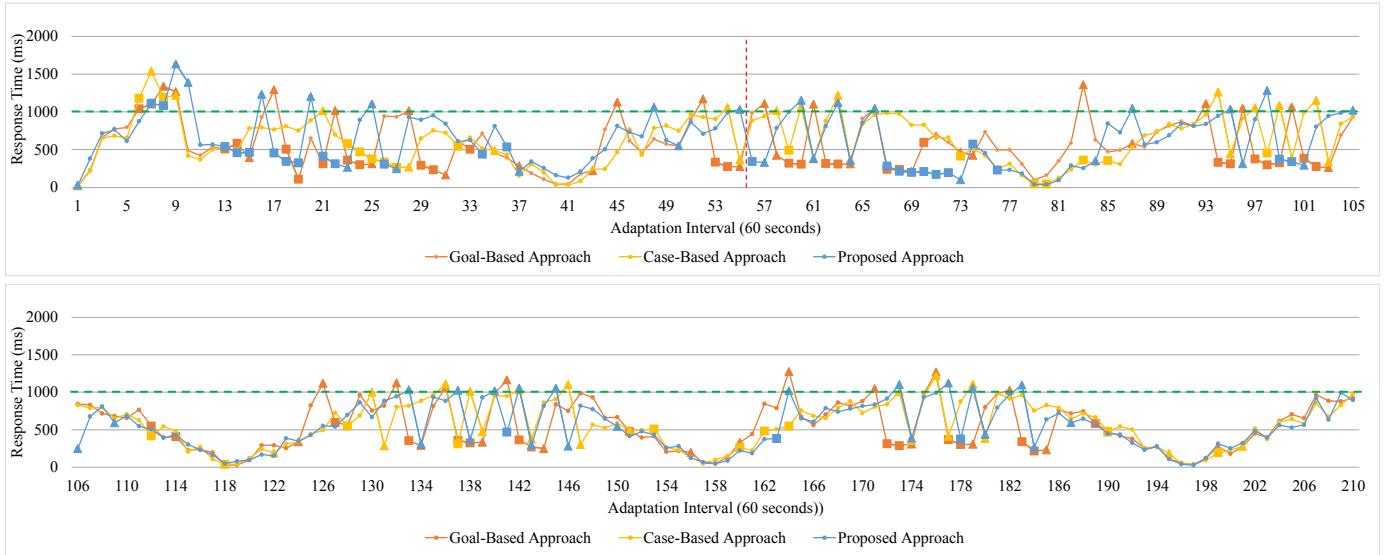


Fig. 6. The Self-Adaptation Process: Adaptation Interval vs. Response Time

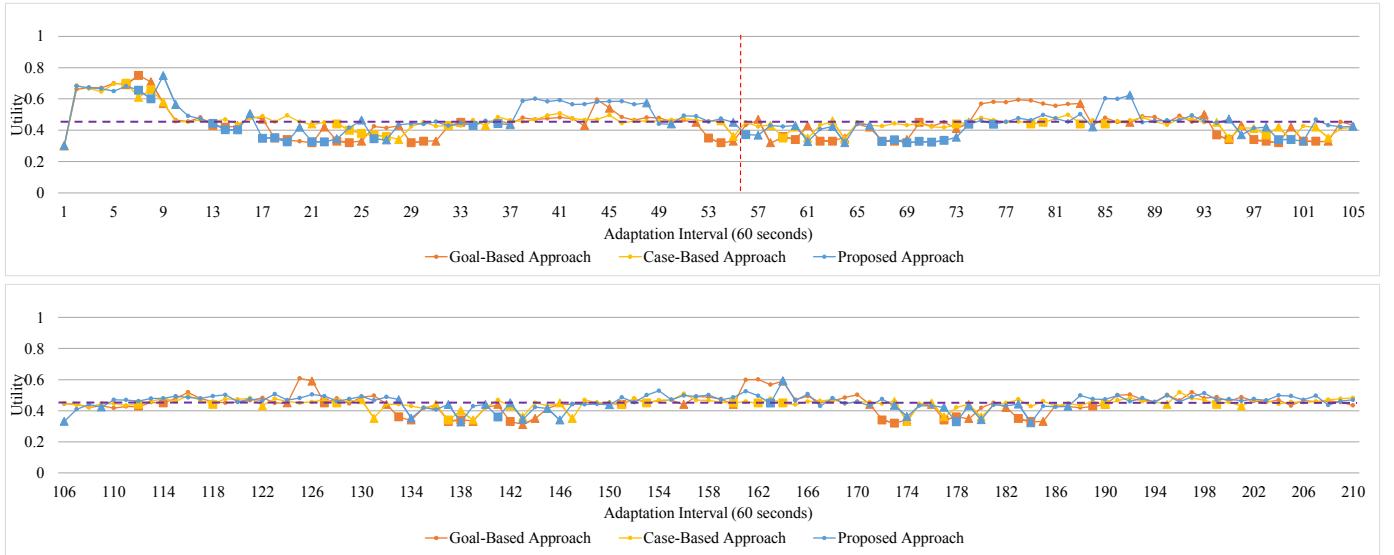


Fig. 7. The Self-Adaptation Process: Adaptation Interval vs. Utility

TABLE III
REASONING MECHANISMS USED IN THE ADAPTATIONS WITH THE PROPOSED APPROACH

| Reasoning Mechanism | Adaptation (#) | Eff. (#) | Rate (%) |
|-----------------------|----------------|----------|----------|
| case-based reasoning | 26 | 23 | 88 |
| goal reasoning | 41 | 17 | 41 |
| synthesized reasoning | 7 | 7 | 100 |

We can see that among the 74 adaptations, 26 adaptations were from case-based reasoning, 41 adaptations were from goal reasoning, and 7 adaptations were from synthesized reasoning. The adaptations from synthesized reasoning had the highest effectiveness rate (i.e. 100%), and the adaptations from goal reasoning had the lowest effectiveness rate (41%). Moreover, the effectiveness of case-based reasoning was improved (recalling that its effectiveness rate in the *Case-Based Approach* is only 65%) because the case base in the *Proposed*

Approach could be incrementally expanded by combining goal reasoning to generate adaptation solutions for new emerging problems (i.e. no similar cases can be found). However, the effectiveness of goal reasoning was decreased (recalling that its effectiveness rate in the *Goal-Based Approach* is 53%) because the *Proposed Approach* would solve a similar problem using case-based reasoning or synthesized reasoning, which reduced the effectiveness rate of goal reasoning.

Furthermore, we divided the 3.5 hours into 7 time slots, and analyzed the distribution of the adaptations that were generated by the three reasoning mechanisms. The results are shown in Figure 8. For example, in the 3rd time slot, there were totally 15 adaptations, and 1 of them was generated by case-based reasoning, 2 of them were generated by synthesized reasoning and 12 of them were generated by goal reasoning.

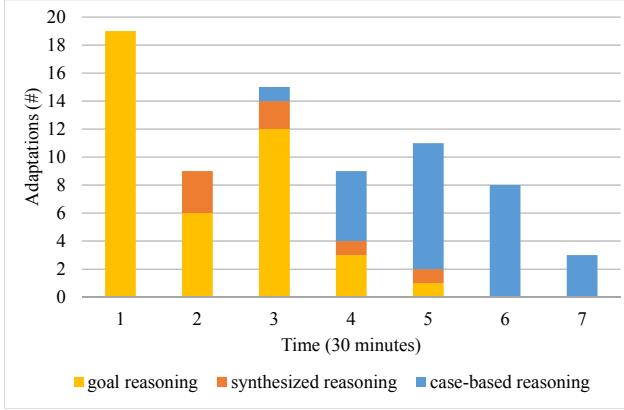


Fig. 8. Distribution of the Adaptations in the Proposed Approach

The trend of the distribution shows that in the beginning the generation of adaptation solutions mostly relied on goal reasoning because the case base was initially empty, and thus too small to handle the emerging problems; then gradually relied more on synthesized reasoning and case-based reasoning because the case base reached a certain size but was still not rich enough; and finally tended to rely on case-based reasoning because the case base became rich enough to handle the most emerging problems.

Specifically, such an adaptation process can be divided into the following three phases:

- **Initialization phase:** Because the case base is initially empty, the adaptation mechanism needs to utilize the rationalism knowledge underlying goal reasoning to help build the case base incrementally and sustainedly.
- **Accumulation phase:** After the initialization phase, there are several cases in the case base. Therefore, similar cases can be retrieved from the case base when there happens a new problem, and case-based reasoning is gradually used. However, because the case base is not rich enough, the retrieved cases may be not very similar to the new problem. As a result, the solutions of the two most similar cases will be synthesized with the solution generated by goal reasoning to get a suitable solution for the new problem. Therefore, both case-based reasoning and synthesized reasoning are gradually used to accumulate adaptations solutions to further enrich the case base.
- **Solving phase:** After the accumulation phase, the case base has been rich enough to handle the most emerging problems. As a result, most adaptation solutions are generated by case-based reasoning, and the goal reasoning and synthesized reasoning are gradually not used.

The above observations from Table III and Figure 8 answers Q2 positively that the proposed approach properly combines goal reasoning and case-based reasoning for producing effective adaptation solutions. Specifically, when no similar cases can be found by case-based reasoning, either goal reasoning or synthesized reasoning can be utilized to produce effective adaptation solutions and thus enrich the case base. When the case base is rich enough, case-based reasoning is utilized to

provide more precise adaptation decisions. In that sense, the three reasoning mechanisms are complementary to each other.

D. Discussion

It's worth noting that the case-based approach implemented in the comparative evaluation uses goal reasoning to construct the original case base. Without this kind of automatic learning of original cases, it is usually hard for case-based reasoning approaches to prepare a comprehensive set of original cases for a self-adaptive system.

Our approach uses a few constants and similarity thresholds, which are usually application-specific and can be specified by experts. Specifically, the setting of α in Algorithm 1 and the changing pace of control variables depends on the tradeoff between tolerance of system disturbance and effectiveness of adaptations. If it is set too large, goal reasoning will produce radical adaptations, probably leading to system disturbance. Otherwise, if it is set too small, goal reasoning will produce less effective adaptations. The setting of $threshold_h$ and $threshold_l$ affects the quality of the reused case in case-based reasoning. If they are set too large, more cases that are not very similar to the problem will be taken into account, which will worsen the solution from case-based reasoning. Otherwise, if they are set too small, less cases with higher similarity will be referenced, which limits the effectiveness of case-based reasoning. In the future, we will conduct sensitivity analysis of these parameters to empirically evaluate their impacts on the effectiveness of our approach.

Last but not the least, ineffective cases may also be helpful for case-based reasoning for self-adaptation. Just as the positive experiences provided by effective cases, negative experiences contained in ineffective cases may also provide useful hints for future adaptation decision making, e.g. to avoid applying an adaptation solution that has been proved to be ineffective. Therefore, it is worthwhile to take such ineffective cases into account to see whether they can provide useful experiences for runtime adaptations.

V. RELATED WORK

Requirements-driven self-adaptive systems are a kind of requirements-aware systems. In such systems, runtime requirements models are used to reason about the satisfaction levels of requirements and to support adaptation decisions [1]. Baresi et al. [2] propose an approach that introduces fuzzy and adaptive goals to embed adaptation countermeasures. Wang et al. [3] propose a self-repairing approach that uses goal reasoning to select a best system configuration. Dalpiaz et al. [4] enrich goal models with context-dependent goal decompositions, goal commitments with time limits, and domain assumptions in their self-reconfiguration architecture. Based on the enriched goal models, their approach monitors for and diagnoses runtime failures by comparing monitored behaviour of a system to expected and allowed behaviours. Peng et al. [5] propose a self-optimization approach that uses a preference-based goal reasoning procedure to reason about optimal goal configurations based on tuned preference ranks by a feedback

controller. Chen et al. [14] propose a requirements-driven self-optimization approach to achieve survivability assurance for Web systems. And their another work [15] proposes a self-adaptation approach that combines requirements and architectural adaptations with model transformation techniques. Salehie et al. [16] propose a requirements-driven approach to support adaptive security in order to protect variable assets at runtime. Fu et al. [17] propose a goal-based monitoring and self-repairing framework for socio-technical systems, which reasons about runtime failures and repairing solutions based on goal state machines and their interactions. Bencomo et al. [18] propose an approach that maps the goal models into Dynamic Decision Networks, and uses the Dynamic Decision Networks to automatically make the best decisions for self-adaptive systems.

These requirements-driven self-adaptation approaches depend on runtime requirements models (usually goal models) to reason about requirements violations and adaptation decisions. The relations among system behaviours, environmental parameters and requirements, serving as the basis of runtime reasoning, are assumed to be clearly understood and modelled. Different from our approach, these approaches do not learn the complex and changing relations from past experiences.

Several attempts have been made to apply case-based reasoning in self-adaptive systems. Montani et al. [9] propose an approach that uses case-based reasoning to diagnose runtime failures of software systems and provide remediation solutions. McSherry et al. [7] propose a hypothesis-driven approach than can diagnose runtime failures and provides recovery suggestions by asking questions and explaining the relevance of the questions to users. Khan et al. [8] propose a self-adaptation approach using case-based reasoning, which can restrict the size of the case base without harming accuracy. These approaches depend on an original case base, which is usually constructed manually by humans based on historical system operation records or human expertise. They cannot provide a proper adaptation solution if there are no similar cases for a given adaptation problem in the case base. By contrast, our approach can provide proper (may not be perfect) adaptation solutions with goal reasoning when no similar cases are available.

VI. CONCLUSIONS

Requirements-driven self-adaptation approaches can make proper adaptation decisions by reasoning over runtime requirements models, but cannot learn from past experiences the complex relations among system behaviours, environmental parameters and requirements.

In this paper, we have proposed an improved requirements-driven self-adaptation approach that combines goal reasoning and case-based reasoning. In our approach, past experiences of successful adaptations are retained in a case base to be reused for future adaptation problems. Both the current system behaviour and the adaptation solution of an adaptation case are specified by goal configurations consisting of specific alternatives for variation points and values for control variables. The

approach reasons over a predefined requirements goal model to provide adaptation solutions when no similar cases are available. The effectiveness of the approach has been evaluated by comparing with a goal reasoning approach and a case-based reasoning approach.

Our future work will investigate some key elements of the approach such as the selection of environmental parameters and the strategy of adaptation solution synthesis, as well as the influence of different thresholds settings. We also consider to apply the proposed approach in more complex Web-based systems to further evaluate its effectiveness and scalability.

VII. ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China under Grant No. 61361120097, National High Technology Development 863 Program of China under Grant No. 2013AA01A605.

REFERENCES

- [1] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein, “Requirements-aware systems: A research agenda for RE for self-adaptive systems,” in *RE*, 2010, pp. 95–103.
- [2] L. Baresi, L. Pasquale, and P. Spoletini, “Fuzzy goals for requirements-driven adaptation,” in *RE*, 2010, pp. 125–134.
- [3] Y. Wang and J. Mylopoulos, “Self-repair through reconfiguration: A requirements engineering approach,” in *ASE*, 2009, pp. 257–268.
- [4] F. Dalpiaz, P. Giorgini, and J. Mylopoulos, “An architecture for requirements-driven self-reconfiguration,” in *CAiSE*, 2009, pp. 246–260.
- [5] X. Peng, B. Chen, Y. Yu, and W. Zhao, “Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop,” *Journal of Systems and Software*, vol. 85, no. 12, pp. 2707–2719, 2012.
- [6] A. Aamodt and E. Plaza, “Case-based reasoning: Foundational issues, methodological variations, and system approaches,” *AI Communications*, vol. 7, no. 1, pp. 39–59, 1994.
- [7] D. McSherry, S. Hassan, and D. Bustard, “Conversational case-based reasoning in self-healing and recovery,” in *ECCBR*, 2008, pp. 340–354.
- [8] M. J. Khan, M. M. Awais, and S. Shamail, “Enabling self-configuration in autonomic systems using case-based reasoning with improved efficiency,” in *ICAS*, 2008, pp. 112–117.
- [9] S. Montani and C. Anglano, “Achieving self-healing in service delivery software systems by means of case-based reasoning,” *Applied Intelligence*, vol. 28, no. 2, pp. 139–152, 2008.
- [10] J. Mylopoulos, L. Chung, and B. Nixon, “Representing and using nonfunctional requirements: A process-oriented approach,” *IEEE Transactions on Software Engineering*, vol. 18, no. 6, pp. 483–497, 1992.
- [11] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, “Reasoning with goal models,” in *ER*, 2002, pp. 167–181.
- [12] V. E. S. Souza, A. Lapouchian, and J. Mylopoulos, “System identification for adaptive software systems: A requirements engineering perspective,” in *ER*, 2011, pp. 346–361.
- [13] J. L. Kolodner, “Improving human decision making through case-based decision aiding,” *AI Magazine*, vol. 12, no. 2, p. 52, 1991.
- [14] B. Chen, X. Peng, Y. Yu, and W. Zhao, “Are your sites down? Requirements-driven self-tuning for the survivability of Web systems,” in *RE*, 2011, pp. 219–228.
- [15] B. Chen, X. Peng, Y. Yu, B. Nuseibeh, and W. Zhao, “Self-adaptation through incremental generative model transformations at runtime,” in *ICSE*, 2014, pp. 676–687.
- [16] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh, “Requirements-driven adaptive security: Protecting variable assets at runtime,” in *RE*, 2012, pp. 111–120.
- [17] L. Fu, X. Peng, Y. Yu, J. Mylopoulos, and W. Zhao, “Stateful requirements monitoring for self-repairing socio-technical systems,” in *RE*, 2012, pp. 121–130.
- [18] N. Bencomo and A. Belaggoun, “Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks,” in *REFSQ*, 2013, pp. 221–236.

TiQi: Towards Natural Language Trace Queries

Piotr Pruski, Sugandha Lohar, Rundale Aquanette, Greg Ott, Sorawit Amornborvornwong,
Alexander Rasin, and Jane Cleland-Huang

School of Computing

DePaul University, Chicago, IL, 60604, USA

ppruski@gmail.com, slohar@cs.depaul.edu, arasin@cs.depaul.edu, jhuang@cs.depaul.edu

Abstract—One of the surprising observations of traceability in practice is the under-utilization of existing trace links. Organizations often create links in order to meet compliance requirements, but then fail to capitalize on the potential benefits of those links to provide support for activities such as impact analysis, test regression selection, and coverage analysis. One of the major adoption barriers is caused by the lack of accessibility to the underlying trace data and the lack of skills many project stakeholders have for formulating complex trace queries. To address these challenges we introduce TiQi, a natural language approach, which allows users to write or speak trace queries in their own words. TiQi includes a vocabulary and associated grammar learned from analyzing NL queries collected from trace practitioners. It is evaluated against trace queries gathered from trace practitioners for two different project environments.

Index Terms—Traceability, Queries, Speech Recognition, Natural Language Processing

I. INTRODUCTION

Software traceability is a critical component of the software engineering process, especially in regulated industries such as avionics, medical devices, and transportation [14]. As a result, developers often invest significant cost and effort creating trace links in order to meet certification or compliance requirements. Unfortunately, despite the significant investment, trace links are often underutilized for supporting activities such as impact analysis, coverage analysis, and test regression selection. There are many possible reasons for this. In some cases, trace links are created late in the software development life-cycle process and are therefore not available for use during earlier development phases [14]. In other cases, lack of tool support, poor understanding of the underlying trace schema, or lack of skills needed to formulate useful trace queries all contribute to making trace links inaccessible to project stakeholders [25], [12]. In particular, there are several documented examples of practitioners who have struggled to generate adequate trace queries [12], either because they lack proficiency in SQL or XQuery, or simply because the available trace information is ill-defined and inaccessible [14].

The problem is exacerbated by the fact that trace queries are often quite complex and cut across many different artifacts. For example, the “producibility” report commissioned by the US Department of Defense [19], and released in 2010, identified traceability as a critical challenge that needs to be addressed in order to provide better support for ongoing evolution of software across an extensive set of artifact types that include

requirements, architectural documents, design, code, and test cases. Our goal is to make traceability information easily accessible to project stakeholders by developing a natural language (NL) interface and supporting tool which allows users to express complex queries using both written and spoken NL.

The idea of NL database queries is far from new. NL query solutions have been developed at differing levels of success from as far back as the 1970s and 80s [24]. While the field stagnated for many years as researchers refocused on developing NL interfaces for less structured sources of information [4], there are several factors which make the database interface problem a compelling one to revisit for traceability purposes today.

First, raw trace data is becoming increasingly available in software projects. There are a number of reasons for this including the fact that regulated industries insist on traceability [14], integrated development environments such as Jazz produce links as a natural byproduct of the development process, and advances in state of the art tracing techniques have matured to the extent that they can be used to instrument the project environment in order to *capture* links or to *generate* trace links through the use of information retrieval techniques [6], [1]. Second, the current ubiquitous move towards mobile computing creates a compelling reason to invest in more flexible and accessible trace usage methods, especially voice activated ones. Third, advances in speech recognition software make speech interfaces a viable option, and finally, previous studies of NL queries for general databases have shown that natural language queries are simpler and more succinct to create than their SQL counterparts [24], suggesting that NL solutions could produce a viable solution to the trace usage adoption barrier.

While a variety of NL approaches exist for issuing general database queries, it is widely accepted that for an NL query language to be effective, it must be supported by a domain specific model [16]. To the best of our knowledge nobody has yet attempted to discover the vocabulary and grammar of queries in the traceability domain or to build an associated NL interface. To address this gap, we present TiQi, a framework and tool supported by a traceability domain model and algorithms capable of transforming naturally worded trace queries into executable SQL statements. The primary contributions of this paper are therefore twofold. First we make advances in

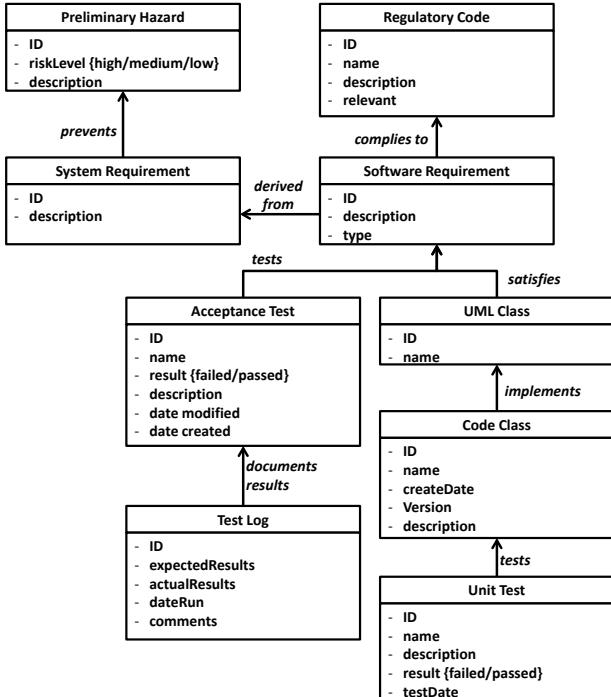


Fig. 1: A Sample Traceability Information Model

learning the vocabulary of a domain model for supporting natural language trace queries, and second we design and implement a NL processing engine capable of transforming NL trace queries into relevant SQL statements.

The remainder of this paper is laid out as follows. Section II discusses existing approaches for issuing trace queries and presents existing NL database query techniques. Section III presents our technique for constructing a domain model, and Section IV describes our TiQi process for transforming a NL query to SQL. In Section V we describe a series of experiments that were conducted to evaluate TiQi. Section VI describes threats to validity, and finally Section VII discusses our findings and proposes future enhancements to TiQi.

II. TOWARDS NATURAL LANGUAGE TRACE QUERIES

In practice, trace queries are issued against an available dataset of artifacts and trace links. Ideally these are documented in a *Traceability Information Model* (TIM) [14] where *artifact types* and their properties are represented as classes and attributes, and *permitted trace links* are represented as semantically typed associations. A TIM can be used to strategically plan the traceability for a project, by serving as a guide for instrumenting the project environment to enable the creation, maintenance, and use of the planned trace links. It can also be reverse engineered from existing datasets using simple parsing tools [21].

Furthermore, the TIM provides useful support for trace queries, as it clearly depicts the artifacts and trace links available for querying purposes. For example, the TIM in Figure 1 allows a project stakeholder to issue a query such as “list all relevant regulatory codes not covered by at least

one software requirement”, or “return a list of hazards which have failed unit test cases associated with them.”

From a physical perspective, as trace links often represent many-to-many dependencies, the trace paths (i.e. associations between artifact types) must be treated as first-class citizens in the underlying database schema. As a result, each artifact, and each association, is represented as a distinct table. Assuming a naming convention in which the trace matrix that establishes links between two artifacts *A* and *B* is called *TM_A_B* and captures each trace link as a pair of IDs taken from *A* and *B* respectively, then the first query could be specified in SQL as follows:

```
SELECT DISTINCT regulatory_code.*  
FROM ((regulatory_code LEFT OUTER JOIN  
tm_regulatory_code_software_requirement ON  
(regulatory_code.id=tm_regulatory_code  
_software_requirement.regulatory_code_id))  
LEFT OUTER JOIN software_requirement ON  
(software_requirement.id=tm_regulatory_code  
_software_requirement.software_requirement_id))  
WHERE regulatory_code.relevant='true'  
AND software_requirement.id IS NULL;
```

Furthermore, a typical software project stores artifacts in a variety of file formats, databases, and proprietary case tools. We therefore cannot assume that artifacts and trace links are nicely stored in a database, and neither can we assume that a complete and correct set of trace links are available. Instead, a traceability environment is more likely to involve a layered approach [13] in which the TIM serves as a logical schema against which all traceability queries are specified; however the elements of the TIM are mapped to physical sources, and executing a query requires the additional step of dynamically retrieving data from those sources in order to populate the database. Tools such as Poirot and RETRO [6] which utilize trace-retrieval techniques to generate candidate trace links upon demand could also potentially be used to service the SQL trace queries generated by TiQi.

A. Related Work: Trace Query Techniques

Trace queries can be issued in a number of different ways. Maeder et al. developed the Visual Trace Modeling Language (VTML) which represents queries as a set of filters applied to a structural subset of the TIM [12]. A VTML query is therefore composed of a connected subset of the artifacts and trace types defined in the TIM, as well as a set of associated filter conditions. These filters are used to eliminate unwanted artifacts or to define the data to be returned by the trace query. Studies conducted with human analysts showed that VTML queries were easier to read and to write than SQL queries. One of their primary advantages is that they allowed the information of the trace matrices to be abstracted away, so that the human user could specify most queries in terms of visible elements of the TIM. Störrle [23] presented the Visual Model Query Language (VMQL), which is similar to, but less expressive than VTML.

Maletic and Collard [15] describe a Trace Query Language (TQL) which can be used to model trace queries for artifacts

represented in XML format. TQL specifies queries on the abstraction level of artifacts and links and hides low-level details of the underlying XPath query language through the use of extension functions. Nevertheless, TQL queries are non-trivial for users without knowledge of XPath and XML to understand. Zhang et al. [26] describe an approach for the automated generation of traceability relations between source code and documentation. They use ontologies to create query-ready abstract representations of both models. The Racer query language (nRQL) is then used to retrieve traces; however nRQL's syntax requires users to have a relatively strong mathematical background.

Guerra et al. [5] build models that describe how modeling languages are inter-related. They transform existing traceability data within a model into a different representation (e.g. a table) against which standard trace queries can be issued. Their approach also requires advanced skills.

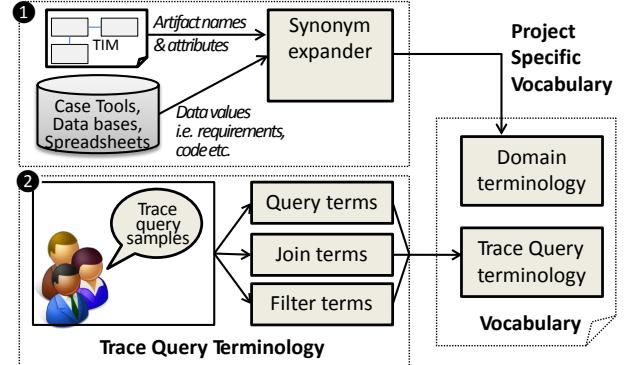
In addition to textual approaches such as SQL, graphical query languages have been proposed and commercially offered in the database domain for a long time. The PICASSO approach by Kim et al. [10] represents one of the earliest graphical query languages and was built on top of the universal relation database system System/U. Visual SQL by Jaakkola et al. [7] translates all features of SQL into a graphical representation similar to entity relationship models and UML class diagrams. Furthermore, a variety of commercial and open source tools provide graphical support for the specification of queries (e.g., Microsoft Visual Studio™, Microsoft Access™, Active Query Builder and Visual SQL Builder).

However, all of these techniques, whether designed specifically for tracing purposes, or for more general database queries require some degree of technical expertise. In contrast, TiQi is designed to accept queries from stakeholders who have no formal technical training in either UML or SQL. The underlying trace query domain model and the TiQi engine is designed to translate higher level concepts and domain-specific jargon into executable trace queries.

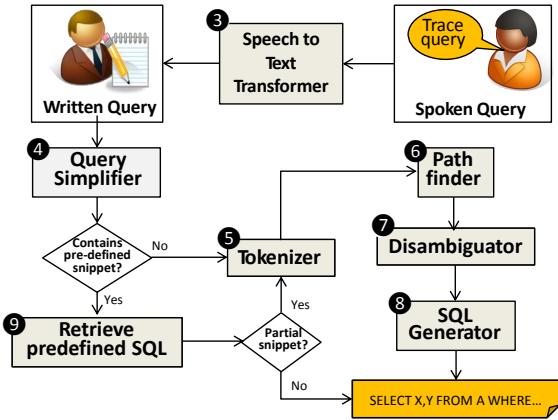
B. Related Work: Natural Language Queries

There are several commercial products that claim to provide NL database interfaces. For example, PrimeQue allows a user to interact with a database by following a series of prompts. The user first builds a semantic map of the dataset, and the tool then uses this map to help the user construct a NL query using a query-by-example style interface [27]. However, such solutions severely constrain the language of the query and therefore do not support truly natural language queries. Other general NL query techniques are primarily limited to the language extracted from the structure of the database i.e. its table names, attribute names etc, and therefore are also rather restrictive [8]. Recently tools such as *EasyAsk Quiri* and Microsoft's *Power BI for Office 365* have demonstrated the viability of using speech interfaces to issue business intelligence queries against database schemas.

In a seminal discussion on the notion of general versus domain specific NL query languages [22], Shwartz provided an



(a) Domain specific vocabulary is extracted from the Traceability Information Model and associated data, while traceability jargon is identified in advance and reused across projects.



(b) A Natural Language Query initiated either as speech or in written form is transformed into SQL

Fig. 2: The TiQi Process

interesting example from the petrochemical domain in which he compared four very similar queries: (1) Show oil wells from 1970 to 1908, (2) show oil wells from 7000 to 8000, (3) show oil wells from 1 to 2000, and finally (4) show oil wells from 40 to 41 and 80 to 81. To an outsider, these queries appear very similar in nature. However, a domain expert would infer that the first query refers to dates, the second to well depth, the third to map depth, and the fourth to well number. A general database system would have little hope of understanding these concepts unless units of measure were explicitly stated. Shwartz therefore claims that conceptual domain-specific knowledge is critical for understanding the nuances of queries.

C-Phrase, developed by Minnock et al., allows users to specify queries using a natural language front end and then to execute those queries against a relational database [18]. C-Phrase provides support for customization for a specific domain by following a *name-tailor-define* cycle. *Naming* involves providing names to classes, attributes, and join relationships. For tracing purposes, these are already specified in the TIM. *Tailoring* allows domain specific patterns to be specified and matched. For example, if we applied this to the traceability domain, then the phrase "Is safe for use" might be associated with "all hazards mitigated" or "all mitigating requirements

addressed”. This phase involves analyzing domain related phrases and identifying matches between them. Finally, in the *define* step, the user provides definitions of terms, for example, the word “recent” might be defined to mean “in the past week.” While C-Phrase provides support for this process, it is generally understood that customizing a model for a specific domain can be extremely time-consuming [2]. Nevertheless we believe the effort is worthwhile for the tracing domain, simply because, once created, it will be applicable across an enormously wide spectrum of software and systems engineering projects. Many of the customization concepts from C-Phrase were adopted in TiQi. However, we were unable to deploy the C-Phrase tool due to obfuscation issues.

Another interesting approach proposed by Meng [17] integrates information from the enterprise, database values, user-words, and query cases. The novelty of this approach is that it directly stores entire sample queries and then, instead of attempting to analyze the individual parts of the NL query and to formulate an executable SQL query, it simply finds a match in the large database of stored queries and issues the associated stored query. Query-cases are somewhat similar to the phrases identified in Minnock’s “tailor” stage. The approach we adopted for TiQi merges concepts from both Minnock’s and Meng’s techniques.

III. BUILDING A DOMAIN MODEL

The domain model for a project is derived from three different sources as depicted in Figure 2a. The first is the structure of the TIM, the second is the underlying traceable data, and the third is the language used to express queries in the domain. The terminology in the TIM and the data compose the project specific vocabulary formulated from artifact names and attribute names. In our current model we do not assume that links will be labeled as we rarely see this in practice; however adding this additional information in future versions of TiQi could enhance its accuracy.

A. Project Specific Vocabulary

To identify the basic vocabulary needed to tokenize a NL query into a set of relations, attributes, values, query terms, filters, join commands, negations, and aggregations, a project specific vocabulary is constructed by parsing the text in the TIM and its underlying artifacts. This produces a list of relation names, attributes, and link names extracted from the meta-data of the TIM, and also a list of values extracted from the data in each of the tables. Each value is associated with one or more attributes. In the sample TIM presented in Figure 1, vocabulary items would include terms such as *preliminary hazard* extracted as a table name, *dateRun* extracted as an attribute, *medium* or *armController* extracted as attribute values, and *tests* or *prevents* extracted as link types. These vocabulary terms can be augmented by synonyms identified using a tool such as WordNet.

B. Domain Concepts and Jargon

To discover an initial set of terms, vocabulary, and query jargon used in the traceability domain, we developed an online

TABLE I: A sample of the collected trace queries

| | |
|---|---|
| 1 | How many high level hazards are associated with the security camera? |
| 2 | Let me see the test log for all system requirements that fail their acceptance test |
| 3 | List all hazards for which the most recent unit test case has failed |
| 4 | Are there any orphaned UML Classes? |
| 5 | List all preliminary hazards that have a high level of risk, and which are not covered by a software requirement of type “mitigating” |
| 6 | Show me all unaddressed low level hazards |
| 7 | Retrieve all the usability related software requirements with currently failed acceptance tests |
| 8 | What percentage of relevant regulatory codes have been fully addressed with passed acceptance tests? |

web-collection tool which displayed a series of TIMs and then asked trace users to think of five useful trace queries for each TIM and to formulate associated NL queries. Participants were encouraged to use any words and phrases that they wished in order to formulate their queries. We deliberately set this as an open-ended exercise because we did not want to constrain the vocabulary choice of the trace users. We included 8 trace experts in the data collection exercise and collected a total of 100 trace queries issued against two different TIMs. Table I shows eight representative queries collected during this exercise.

Prior work by Meng and Siu [17] proposed automated approaches for extracting grammars for a new domain. Their approach infers a grammar from an unannotated corpora of queries. While such approaches have been shown to be fairly effective, they require a very large corpora of sample queries. Given the limited number of sample queries available to us at this time, we chose to extract the vocabulary manually through systematically analyzing each of the queries to identify *task-related terms*, *join-terms*, and *filter terms*. Each of the observed terms were then mapped into a simple representative term. This step was similar to Minnock’s *tailor* step [18]. Based on the initial sample of 100 queries we mapped 94 phrases and concepts, such as the mapping of *mitigates* to *prevents*.

Based on this simple exercise we identified three different categories of queries that we classified as *solvable*, *ambiguous*, and *intractable*. An *intractable* trace query is one which is unsolvable with respect to the TIM. For example, queries such as “Is the design flexible enough to accommodate new system-requirements?” or “Does the code follow proper object-oriented programming practices?” address interesting questions but are not supported by underlying trace data and are therefore outside TiQi’s scope. Similarly nonsense queries such as “Dude, what’s up?” are also intractable. An *ambiguous* query is one which has more than one reasonable interpretation (even to a human). Finally, a *solvable* query is one which is neither intractable nor ambiguous. It is correctly solved only if TiQi returns the correct information to the user.

IV. THE TIQI MODEL

The overall TiQi process transforms a natural language trace query into an executable SQL statement. Its primary

elements are depicted in Figure 2b. Steps 3-9 are executed each time an NL trace query is issued. In the following sections we describe each of these steps and illustrate them with the running example of the query: *I'd like to see a list of all preliminary hazards for arm movements which are tested by recent unit tests.* The intermediate forms of the query are depicted in Figure 3.

A. Speech to Text

Our TiQi approach includes the option of speaking or writing the NL trace query. All spoken queries are transformed into textual format as a preprocessing step. We investigated several tools for supporting the speech to text step, focusing mainly on CMU Sphinx as a local speech processor, and Google Speech API as a web-based option. After training audio data and generating custom, domain specific language data for Sphinx, the two applications had similar accuracy rates. However, Google's Speech API was more portable, faster and, ultimately, performed better on average when presented with new data, so it was chosen as our speech processor.

B. Query Simplification

The query simplification process parses and tokenizes the query. This process basically involves matching phrases found in the query to specific mapping rules, and then replacing the phrases with the mapped terms. Specific heuristics included phrase-based synonym matching, definition replacements, and the transformation of numbers, durations, times, and dates into standard forms. For example, phrases such as “I'd like to see” and “display every” were mapped to the term *list*, and meanings were defined for words such as “recently”, which was defined to mean “within the past week”. Such definitions must be defined by stakeholders at the project level. Our initial mappings were based upon our analysis of the initial collection of trace queries.

We utilized the Stanford Parser to identify numbers, durations, times, and dates. The parser returned labeled parts of speech which facilitated the extraction of relevant text and its subsequent transformation into a standardized format from which it was possible to generate SQL.

C. Tokenizer

The tokenizer is responsible for identifying key terms in the simplified query. It performs this task by searching the lexicon for relation names, attribute names, attribute values, and link names. In our running example *list* is tagged as a descriptive term. *High* is identified as a value associated with one of three possible attributes: PreliminaryHazard.riskLevel, UMLClass.name, or CodeClass.name. *riskLevel* is recognized as an attribute in the PreliminaryHazard relation. Similarly other terms are all mapped to candidate relations, attributes, or values as depicted in Figure 3.

Clearly there are many ambiguities that arise as a result of the tokenization. For example, it is unclear whether the term *class* means *Code Class* or *UML Class*. Disambiguation is deferred to a later step of the process. In the case that a

word receives no assignment i.e. the word is not found in the vocabulary of the domain either as a basic term, a synonym, or as a mappable definition, then the query is marked as intractable.

D. Path Finder

In many cases, trace queries will be specified only in terms of the source and target artifacts, even though the actual traceability path flows through a set of intermediate artifacts. To create an executable query, we need to explicitly identify the actual path of the trace query.

To accomplish this we conduct a depth-first search to identify the complete set of trace paths between source and target artifacts. It is important to note that while all of our TIMs allowed only a single path between each pair of artifacts, there are sometimes valid reasons for redundant paths to appear in a TIM [14]. For example, a small set of critical requirements might be traced via state transition diagrams to code, while the remaining requirements are traced directly to code. The depth-first search is feasible because of the relatively small size of the TIM. In TIMs with no redundant trace paths a faster approach, such as Dijkstra's shortest path algorithm, can be used [3]. In the case that multiple candidate paths are found, the *disambiguator* must determine which is the correct one.

E. Disambiguator

The primary task of the disambiguator is to attach each value to a single valid attribute and each attribute to a single valid relation. In cases where ambiguity exists i.e. more than one match is viable, its job is to either select the correct mapping or to seek clarification from the user.

Disambiguation is quite complex, and several different techniques have been used in prior work. TiQi therefore adopts a toolbox of prioritized techniques and heuristics and uses them to resolve a variety of different kinds of ambiguities. The extent to which these ambiguities are correctly resolved directly impacts the correctness of the generated SQL queries.

Based on our observations and analysis of sample queries and their SQL implementation, we developed an initial set of heuristic rules. Rules were added systematically as each query was examined and were then refined and tested against other queries in the sample dataset.

Each rule has the ability to filter the candidate results, but does not necessarily identify a single result, therefore it is often necessary to apply rules sequentially until a single table, attribute, or value is identified. In future work we plan to adopt a more probabilistic approach. We list the primary rules here.

1) **Rule 1: No Competition:** If a token maps only to one table, one attribute, or one value element, then that element is selected.

2) **Rule 2: Max-Flow:** Popescu et al. showed that a certain class of ambiguity could be definitively disambiguated through use of the *Max-flow* algorithm [20]. Maximum flow problems are setup as single-source and single-sink networks, and the goal is to maximize the flow through the network. For query

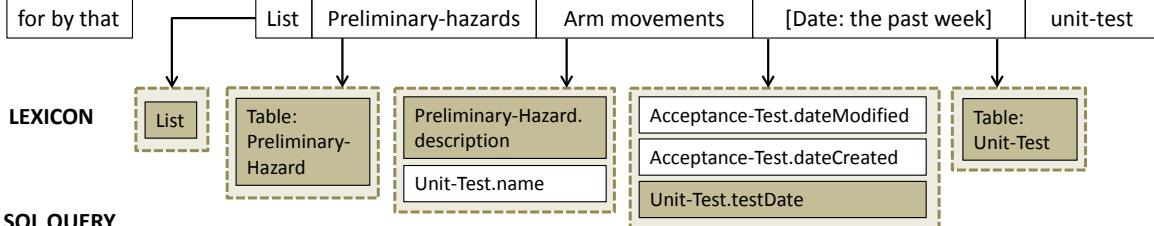
QUERY

I'd like to see a list of all preliminary hazards for arm movements which are tested by recent unit tests.

PRE-PROCESSED QUERY

[List] preliminary-hazard for arm movement [Join] tested by [Date: the past week] unit-test.

SYNTACTIC MARKERS



SQL QUERY

```

SELECT `PreliminaryHazard`.*
FROM `PreliminaryHazard`, `LINKSystemRequirement2PreliminaryHazard`, `SystemRequirement`,
`LINKSoftwareRequirement2SystemRequirement`, `SoftwareRequirement`, `LINKUMLClass2SoftwareRequirement`,
`UMLClass`, `LINKCodeClass2UMLClass`, `UMLCode`, `LINKUnitTest2CodeClass`, `UnitTest`
WHERE `PreliminaryHazard`.`ID` = `LINKSystemRequirement2PreliminaryHazard`.`TargetID` AND
`SystemRequirement`.`ID` = `LINKSystemRequirement2PreliminaryHazard`.`SourceID` AND
`SystemRequirementID`.`ID` = `LINKSoftwareRequirement2SystemRequirement`.`TargetID` AND
`SoftwareRequirement`.`ID` = `LINKSoftwareRequirement2SystemRequirement`.`SourceID` AND
`LINKUMLClass2SoftwareRequirement`.`TargetID` AND `UMLClass`.`ID` = `LINKUMLClass2SoftwareRequirement`.`SourceID` AND
`UMLClass`.`ID` = `LINKCodeClass2UMLClass`.`TargetID` AND `UMLCode`.`ID` = `LINKCodeClass2UMLClass`.`SourceID` AND
`UMLCode`.`ID` = `LINKUnitTest2CodeClass`.`TargetID` AND `UnitTest`.`ID` = `LINKUnitTest2CodeClass`.`SourceID` AND
`UnitTest`.`testDate` >= "03/01/2014" AND `PreliminaryHazard`.`Description` LIKE "%arm movement%";
```

Fig. 3: Steps in the transformation process from a NL Query to an executable SQL query. The query is first simplified and then tokenized through mapping to the lexicon. When alternate mappings are possible, disambiguation occurs through applying a set of heuristic rules and executing the Max-flow algorithm to optimize satisfaction of the results.

disambiguation purposes, a graph is created to contain the candidate database values spoken by the user. The single-source is the system, first level nodes are created for each candidate value's table and column, second level nodes contain each value, which link to the output sink. Edges are established between the first level nodes and their potential database values in the second level, and are assigned capacities of one. The Max-flow algorithm guarantees to identify correct flow paths when conflicts occur, and it can identify when an ambiguous query that does not contain a conflict has been passed to the graph. The Max-flow graph for our example query is shown in Figure 4. All queries processed by our TiQi model are directed through the Max-flow algorithm. In those cases in which it was able to resolve conflicts, the query is modified accordingly, otherwise it is left unchanged.

3) **Rule 3: Pecking Order:** If a name of an item matches more than one element type i.e. table, attribute, and/or a value, then it is assigned first to a table, then to an attribute, and only in the final case to a value.

4) **Rule 4: Compounding Evidence:** When an attribute or value could potentially be associated with multiple candidate tables, for example if two tables share attributes of the same name, then a total weighting is computed based on the number of potential attribute and value matches to each table. The table with the highest total weighting (i.e. the strongest evidence) is selected. For example, it is unclear whether the term *failed*

refers to "UnitTest.result" or "AcceptanceTest.result". However, the direct and unambiguous mapping of the term *unitTest* to the relation *unitTest* provides compounding evidence for selecting *UnitTest.Result*.

5) **Rule 5: Smaller is Better:** If a token matches a value found in more than one attribute, then attributes containing fields with fewer words are selected over those containing fields with more words. For example, if the token "critical" is found in a query and there are two options to map it as a value associated with a "status" attribute (with an average word count of 1 word per record) versus a "description" attribute (with an average word count of 20 words per record), then it is mapped to the "status" attribute.

6) **Rule 6: Neighbors first:** If a token k_1 has been mapped to table T_1 , and there are options to map a second token k_2 to either tables T_2 or T_3 , the distance from T_2 and T_3 to T_1 is measured and the closest table chosen.

Unless rules 1 and 2 result in complete disambiguation of the tokens, we cannot guarantee that the query will be correctly interpreted. However, statistical inferencing techniques can be used in domains for which a large number of queries are available [20]. In these circumstances it would be possible to analyze the use of phrases and terms versus the underlying intent of the query and then to infer the meaning of a query within some confidence interval. We were unable to implement statistical inferencing in our prototype implementation of TiQi

QUERY TOKENS

```
Preliminary-Hazard.Description="Deposits"
Unit-Test.Name="Deposits"
Unit-Test.Name="Connections"
Unit-Test.testDate>="9/1/2013"
```

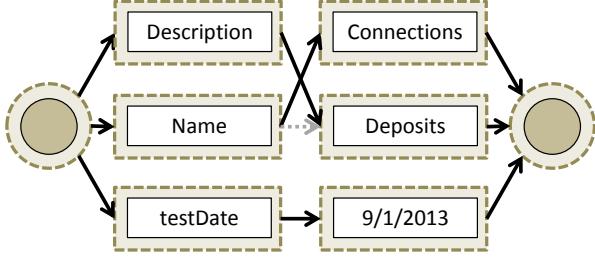


Fig. 4: The Max-flow Graph. Solid black edges indicate links, whereas dashed grey edges indicate that no flow exists.

as we do not yet have sufficient trace queries; however, this approach will be integrated in future versions of TiQi.

F. SQL Generation

Once the incorrect results have been trimmed in the disambiguator, and the conflicts have been resolved, every token in the original query is mapped to one, and only one target artifact. From this, the formal SQL query can be generated as depicted in Figure 3.

V. EVALUATING TiQi

We conducted two different experiments. The first was designed to explore user preference for spoken versus written natural language queries and to evaluate the correctness of the queries. The second directly evaluated TiQi’s ability to accept a range of natural language queries and to transform them into correct SQL statements which returned the desired trace information. We release both datasets used in these experiments, including TIMs, databases, and queries at <http://re.cs.depaul.edu/tiqi/dataset.html>.

A. Comparison of Techniques

We designed a series of experiments to evaluate the extent to which users could effectively create trace queries using SQL, Speech, and written NL. We also explored their preferences for these three techniques. To this end we measured both the time it took users to create the various kinds of queries, and also the quality of the resulting query.

One of the challenges in designing this experiment was in deciding how to elicit similar queries from the user for each of the three query specification techniques. We could not simply describe all the queries in words, because this would suggest the actual solution for the NL queries. Similarly, if we prompted for SQL queries with words, and NL queries using SQL, our experiment would end up evaluating readability of one query-style in tandem with writability of another query-style. Furthermore, we anticipated that the NL queries would deteriorate into “verbal SQL” which was clearly not our intent. We therefore constructed concrete problem scenarios

TABLE II: Experimental design showing the query types elicited by prompt for three different groups of participants

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 |
|---------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Group A | SQL | Txt | Sp | SQL | Txt | Sp | SQL | Txt | Sp |
| Group B | Sp | SQL | Txt | Sp | SQL | Txt | Sp | SQL | Txt |
| Group C | Txt | Sp | SQL | Txt | Sp | SQL | Txt | Sp | SQL |

and asked participants to design a supporting trace query. A web-based tool was developed which displayed a TIM and a related problem scenario to the user, and prompted the user to enter or to record a trace query using one of the three query methods. The TIM contained very standard artifacts including requirements, code, UML classes, acceptance tests, and test logs which any practicing Software Developer would be familiar with. None of the prompts required domain knowledge.

For example, Prompt-1 (P1) stated that “The safety officer is worried that an important requirement *R136* is not correctly implemented. The developer tells him that it is not only implemented but has also passed its acceptance tests. The security officer runs a trace query to confirm this. What is his query?” This prompt is designed to elicit a trace query which either (1) lists all requirements which have not passed their most recent acceptance tests, (2) counts the number of requirements with failed acceptance tests, or (3) simply lists the test status of all requirements. All three of these queries require a trace query that incorporates information from two distinct tables in the TIM.

To reduce the bias introduced by the order in which various query types were elicited, we adopted an interleaved experimental design in which each subsequent participant was assigned to one of three groups (A,B, or C). All three groups received the prompts in the same order, however the type of query i.e. NL-Speech, NL-Text, or SQL, was presented in three different orders as shown in Table II. Subsequently, by the end of the experiment, each prompt had been addressed approximately an equal number of times using each of the three techniques, and the ordering of the techniques was varied across the three groups.

21 people participated in our study. 11 of them classified themselves as traceability experts. Of these, four used traceability in their workplace, and 7 were traceability researchers, well versed in creating and using trace links. The remaining participants were IT professionals, either architects, developers, or project managers, who understood the tenets of traceability but did not use it in the workplace. We provided a basic web-based tutorial on traceability which all participants were required to view before participating in the study.

1) *Query Creation Time*: The first research question evaluated whether users could specify NL queries more quickly than SQL ones. We hypothesized (H1) that the time taken to create natural language trace queries was less than that taken to create SQL trace queries. We recorded the time each participant took to view the scenario prompt and specify the query. Results were aggregated for all participants. The average time taken to create a query for each scenario using

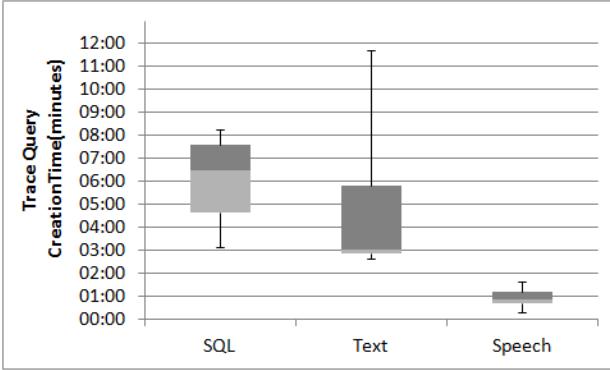


Fig. 5: Trace query creation time for each technique.

each of the three techniques was first computed and then the mean query creation time was calculated for each technique.

Results are displayed in Figure 5 as a Box and Whisker graph and show that NL-Speech queries were the fastest with a mean of 56 seconds, NL-Text came next at 4 minutes and 47 seconds, and SQL was the slowest at 6 minutes and 2 seconds. A t-test failed to reveal a statistically reliable difference between the two means at $\text{Sig}(2\text{-tailed})$ value of 0.308 ($p < 0.05$). However, a second t-test was conducted to compare the mean query creation time for SQL versus NL-Speech and found a statistically significant difference between the two means at $\text{sig}(2\text{-tailed})$ value of 0.0001 ($p < 0.05$).

We also observed that there was a large distribution in the time people took to create NL-Text queries. Although, more than 75% of the queries were created in approximately 6 minutes or less, a few queries took much longer. In these cases the user had tried to recreate the SQL query in spoken form.

2) *Query Correctness*: The second research question evaluated the extent to which each of the three techniques produced correct trace queries. We hypothesized that the correctness of both spoken and written natural language trace queries would be at least as good as that of SQL trace queries.

We evaluated the quality of the generated trace queries based loosely on a framework proposed by Jarke et al. [9] for evaluating structured natural language versus SQL in general database queries. Queries were classified into four categories of *correct*, *minor* substance error, *major* substance error, and *incomplete*. The categories are listed here from [9].

Correct: Statements are formulated correctly and, if executed, could return the correct information. For natural language queries this meant that the query was solvable as defined in Section III-B.

Minor Error: Query output would have been correct, but a minor error is introduced in the statement of the problem. Such errors can be fixed with minor changes.

Major Error: The query is not for the request at hand but for a different one.

Incomplete: Either the solution is incomplete or no attempt was made to create a query.

We assessed each query manually as we did not want the effectiveness of the NL-Queries to be judged according to the

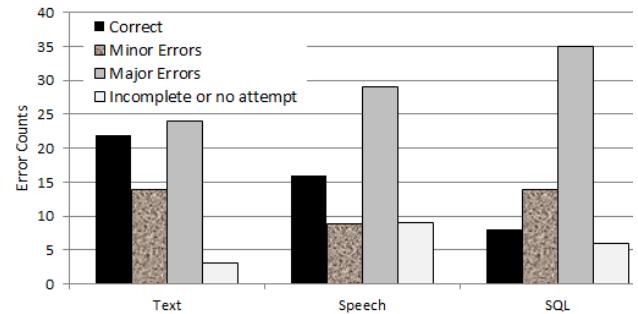


Fig. 6: Correctness of Speech, Text, and SQL queries

current ability of TiQi. Each query was therefore encoded manually by two members of the team following a clearly defined set of prescribed rules. When disagreements occurred a third team member was consulted. For example, for the NL-approach, we encoded queries as major errors if they included entire paragraphs of text instead of succinct queries. For each of the three query techniques we calculated the percentage classified into each category. As reported in Figure 6, only eight SQL queries were categorized as completely correct compared to 16 for Speech and 22 for Text. Both SQL and Text had 14 minor errors, while Speech had only 9. Merging these two categories into a ‘basically correct’ one results in 36 for Text, 25 for Speech, and only 22 for SQL. As we report later in this paper, SQL is far less forgiving of minor errors than the two natural language approaches. At the other end of the spectrum, all three techniques had at least 24 of their queries categorized as having major substance errors. We provide insights into this when we discuss the users feedback in Section V-A3.

3) *User Perception*: We designed a post-test survey to discover the participants’ perspective on the three techniques. In particular we were interested in their preferred query technique, and also in the perceived difficulty of the three approaches.

The first question asked each participant to rate the three approaches of creating trace queries in terms of difficulty on a scale of 1 to 5, with 1 being the easiest and 5 being the most difficult. Results are reported in Figure 7 and show that in general more people found it difficult to formulate queries in SQL than in either Speech or Text. In fact, 15 of 21 participants assigned SQL a score of four or higher on the difficulty scale, in comparison to only three for speech and one for text. Conversely 12 people assigned a difficulty score of two or lower to Speech, 12 to text, but only two to SQL. We can therefore clearly see that in general people found both Speech and Text queries simpler to formulate than SQL ones, with a slight preference for text.

The second question asked each participant to select their preferred query technique. Overall results showed that 14 preferred NL-Text, four preferred SQL, and only three preferred speech. We also examined preferred query techniques for traceability experts versus non experts. Interestingly out of the 10 non-experts seven preferred Text and 3 preferred SQL. None of them registered a preference for Speech. In the

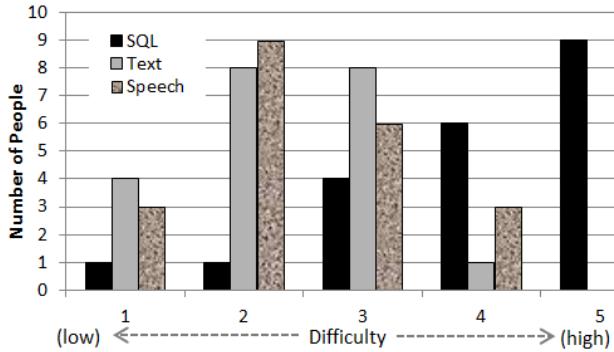


Fig. 7: User perceptions of Query Difficulty

case of the traceability experts, three people out of 11 said that they preferred NL-speech, seven preferred NL-Text, and only one preferred SQL. These results are quite interesting and show strong support for the use of NL trace queries. 2 Feedback from participants highlighted several issues. Some people preferred written NL trace queries over spoken ones because they felt that it gave them time to plan the query more effectively. They were also concerned that the speech recognition software might not recognize their voice efficiently and that this would impact the correctness of the query. On the other hand other participants liked the spoken approach as they felt that they didn't need to worry about the finding the correct syntax to frame a query. Many participants commented that creating trace queries in SQL was very complex, especially because they needed to carefully handle the joins in order to establish links through the underlying trace matrices. On the other hand those who preferred SQL over NL approach felt that with SQL they could state the queries more precisely without worrying about potential ambiguities.

B. Natural Language to SQL Transformations

The second experiment focused on the ability of TiQi to transform written NL queries to executable SQL that returned the targeted information. For these purposes we used the prototype TiQi tool we developed in C#. The current implementation of TiQi includes features such as path finding, token disambiguation, date and time recognition and standardization, definitions and synonyms, simple negation of attributes, and the transformation of structured, tokenized sentences to SQL. Results were evaluated by executing the generated SQL trace queries and then inspecting both the returned data and the original SQL statements. These inspections were performed by three members of our team. In cases for which a textual query could be interpreted in more than one way, we accepted any valid interpretation of the query. Experiments were conducted against the following two datasets.

1) *Isolette Data set*: The Isolette TIM includes eight different artifacts, namely: hazards, faults, environmental assumptions, system requirements, design, code, test cases, and test results connected through six unique trace paths. Together these artifacts contain a total of 26 attributes in addition to IDs. The actual data was primarily extracted from a documented case study about a safety-critical Isolette system [11]. 70 NL

trace queries were developed. Queries were created by five different software engineers and included a variety of jargon, dates, times, negation, and included non-trivial queries which could only be addressed by traversing multiple artifacts and trace matrices.

2) *Easy Clinic*: The Easy-Clinic TIM includes 7 different artifacts, namely: HIPAA-goals, requirements, design, code, unit and acceptance test case and test log connected through six unique trace paths. Together these artifacts contain a total of 17 attributes in addition to IDs. Functional Requirements, code classes and test cases were derived directly from Easy-Clinic data (available from CoEST.org). HIPAA goals were taken from the US HIPAA Technical Safeguards. All other artifacts were created by one of the researchers on this project for purposes of the project. As Easy-Clinic contains a mix of English and Italian terms, we translated Italian terms to English. 40 NL trace queries were developed.

C. Results

Results were evaluated by running each trace query through TiQi and then reviewing both the generated SQL and the produced data result. Each result was marked as either correct or incorrect. We report results for the number of completely correct queries versus incorrect ones for each dataset in Table III. We also report results for the subset of queries for which all necessary TiQi functions have been implemented. More specifically, we exclude queries which required aggregation and identification of missing artifacts (e.g., not yet tested).

TABLE III: SQL queries generated from NL Text

| All Queries | Correct | Incorrect | % Correct |
|-------------------|---------|-----------|-----------|
| Isolette | 49 | 21 | 70.0% |
| Easy Clinic | 25 | 15 | 62.5% |
| Supported Queries | Correct | Incorrect | % Correct |
| Isolette | 49 | 7 | 87.5% |
| Easy Clinic | 25 | 9 | 73.53% |

Results for the Isolette dataset were very promising with 70% of the queries transformed correctly. Mistakes primarily involved the inclusion of extra tables and constraints. On the other hand, only 62.5% of the Easy-Clinic queries were correctly transformed. An initial analysis suggested a double edged problem which included more verbose queries and some ambiguous attribute names. Nevertheless both data sets represent realistic domains and will serve as baselines for future improvements.

VI. THREATS TO VALIDITY

External validity evaluates the ability of the approach to generalize well out of samples used in the experiments. One threat to validity is introduced by the fact that we only evaluated our approach against two different TIMs and a limited set of trace queries many of which were created by members of our team. While our TIMs were non-trivial in size and complexity, they did not contain multiple hierarchies of systems and subsystems which is common in some large

system engineering projects. On the other hand, the TIMs represented a variety of artifact types and attributes and were representative of numerous software projects. A second threat is introduced by the fact that our vocabulary and rules were built from a relatively small sample of trace queries. It is clearly important to extend the sample size in the future so that we can construct a more robust transformation process.

Construct validity refers to whether the dependent and independent variables are suitable for evaluating the hypothesis, and therefore of answering the stated research questions. In our user study, the primary independent variable was the query method, i.e. spoken NL, written NL, or SQL, while, in two of the evaluations, the dependent variables were based on a rubric and involved human assessment. To mitigate bias we carefully defined rules for placing queries into each of the categories and these rules were systematically followed.

Finally, internal validity reflects the extent to which a study minimizes systematic error or bias, so that a causal conclusion can be drawn. We mitigated systematic error by randomly assigning participants to groups, and then adopting an interleaving approach in which different groups were given tasks in different orders for each of the TIMs.

VII. CONCLUSION

The work described in this paper transforms natural language trace queries into SQL. The experiments that we conducted and the data we gathered show that the majority of users had a preference for written NL queries over either speech or SQL, and that in fact such queries were written more accurately and were perceived as being less difficult than SQL. These results support the notion that a NL interface can potentially be useful for helping project stakeholders utilize project trace data in a meaningful way. Our second experiment demonstrated the viability of transforming trace queries into executable SQL. Nevertheless there is still a large amount of work to be done in order for TiQi to consistently and accurately handle diverse and complex queries.

In future work we plan to collect trace queries from practitioners working in safety critical domains so that we can construct a more extensive trace query domain model, with a more complete set of disambiguators and transformation rules. Furthermore we plan to evaluate our approach against more complex trace queries and build supporting tools for engaging users interactively in the disambiguation process.

VIII. ACKNOWLEDGMENTS

This work was partially funded under US National Science Foundation Grants CCF 1319680.

REFERENCES

- [1] N. Ali, Y.-G. Guéhéneuc, and G. Antoniol. Trustrace: Mining software repositories to improve the accuracy of requirement traceability links. *IEEE Trans. Software Eng.*, 39(5):725–741, 2013.
- [2] I. Androutsopoulos and G. Ritchie. Database interfaces. In *Handbook of Natural Language Processing*, pages 209–240. Marcel Dekker Inc., 2000.
- [3] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, editors. *Introduction to Algorithms (Second ed.)*. MIT Press and McGrawHill, 2001.
- [4] M. H. Göker, C. A. Thompson, S. Arajärvi, and K. Hua. Connecting people with questions to people with answers. *KI*, 21(4):23–26, 2007.
- [5] E. Guerra, J. de Lara, D. Kolovos, and R. Paige. Inter-modelling: From theory to practice. In D. Petriu, N. Rouquette, and Ø. Haugen, editors, *Model Driven Engineering Languages and Systems*, volume 6394 of *Lecture Notes in Computer Science*, pages 376–391. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16145-2.
- [6] J. H. Hayes, A. Dekhtyar, S. K. Sundaram, E. A. Holbrook, S. Vadlamudi, and A. April. Requirements tracing on target (retro): improving software maintenance through traceability recovery. *ISSE*, 3(3):193–202, 2007.
- [7] H. Jaakkola and B. Thalheim. Visual sql – high-quality er-based query treatment. In M. Jeusfeld and Ó. Pastor, editors, *Conceptual Modeling for Novel Application Domains*, volume 2814 of *Lecture Notes in Computer Science*, pages 129–139. Springer Berlin / Heidelberg, 2003. 10.1007/978-3-540-39597-3.
- [8] M. Jarke, J. Krause, and Y. Vassiliou. Studies in the evaluation of a domain-independent natural language query system. In *Cooperative interfaces to information systems*, pages 101–130. Springer, 1986.
- [9] M. Jarke, J. Krause, Y. Vassiliou, E. A. Stohr, J. A. Turner, and N. H. White. Evaluation and assessment of a domain-independent natural language query system. *IEEE Database Eng. Bull.*, 8(3):34–44, 1985.
- [10] H.-J. Kim, H. F. Korth, and A. Silberschatz. Picasso: a graphical query language. *Software – Practice and Experience*, 18:169–203, March 1988.
- [11] D. L. Lempia and S. P. Miller. Requirements engineering management handbook. *National Technical Information Service (NTIS)*, 2009.
- [12] P. Mäder and J. Cleland-Huang. A visual language for modeling and executing traceability queries. *Software and System Modeling*, 12(3):537–553, 2013.
- [13] P. Mäder, O. Gotel, and I. Philippow. Getting back to basics: Promoting the use of a traceability information model in practice. In *Traceability in Emerging Forms of Software Engineering, 2009. TEFSE '09. ICSE Workshop on*, pages 21 –25, may 2009.
- [14] P. Mäder, P. L. Jones, Y. Zhang, and J. Cleland-Huang. Strategic traceability for safety-critical projects. *IEEE Software*, 30(3):58–66, 2013.
- [15] J. I. Maletic and M. L. Collard. Tql: A query language to support traceability. In *TEFSE '09: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, pages 16–20, Washington, DC, USA, 2009. IEEE Computer Society.
- [16] P. McFetridge and C. Groeneboer. Novel terms and cooperation in a natural language interface. In *Knowledge Based Computer Systems*, pages 331–340. Springer, 1990.
- [17] H. H. Meng and K. C. Siu. Semiautomatic acquisition of semantic structures for understanding domain-specific natural language queries. *IEEE Trans. on Knowl. and Data Eng.*, 14(1):172–181, Jan. 2002.
- [18] M. Minock. C-phrase: A system for building robust natural language interfaces to databases. *Data Knowl. Eng.*, 69(3):290–302, 2010.
- [19] National Research Council of the National Academies, The National Academies Press, Washington DC. *Critical Code, Software Productivity for Defense*. 2010.
- [20] A.-M. Popescu, O. Etzioni, and H. A. Kautz. Towards a theory of natural language interfaces to databases. In *IUI*, pages 149–157, 2003.
- [21] P. Rempel, P. Mäder, T. Kuschke, and J. Cleland-Huang. Mind the gap: Assessing the conformance of software traceability to relevant guidelines. In *Intr'l Conf. on Software Engineering (ICSE)*, 2014.
- [22] S. P. Shwartz. Problems with domain-independent natural language database access systems. In *Proceedings of the 20th annual meeting on Association for Computational Linguistics, ACL '82*, pages 60–62, Stroudsburg, PA, USA, 1982. Association for Computational Linguistics.
- [23] H. Störrle. VML: A visual language for ad-hoc model querying. *Journal of Visual Languages & Computing*, 22(1):3–29, 2011.
- [24] Y. Vassiliou, M. Jarke, E. Stohr, J. Turner, and N. White. Natural language for database queries: A laboratory study. *MIS Quarterly*, 7(4):47–61, 1983.
- [25] S. Winkler and J. von Pilgrim. A survey of traceability in requirements engineering and model-driven development. *Software and System Modeling*, 9(4):529–565, 2010.
- [26] Y. Zhang, R. Witte, J. Rilling, and V. Haarslev. An ontology-based approach for the recovery of traceability links. In *3rd Int. Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM 2006)*, Genoa, Italy, October 1st 2006.
- [27] M. Zloof. Query by example. In *Proceedings of the NCC*. AFIPS.

Traceability-Enabled Refactoring for Managing Just-In-Time Requirements

Nan Niu*, Tanmay Bhowmik†, Hui Liu‡, and Zhendong Niu‡

* Department of Electrical Engineering and Computing Systems, University of Cincinnati, USA

† Department of Computer Science and Engineering, Mississippi State University, USA

‡ School of Computer Science and Technology, Beijing Institute of Technology, China

nan.niu@uc.edu, tb394@msstate.edu, {liuhui08, zniu}@bit.edu.cn

Abstract—Just-in-time requirements management, characterized by lightweight representation and continuous refinement of requirements, fits many iterative and incremental development projects. Being lightweight and flexible, however, can cause wasteful and procrastinated implementation, leaving certain stakeholder goals not satisfied. This paper proposes traceability-enabled refactoring aimed at fulfilling more requirements fully. We make a novel use of requirements traceability to accurately locate where the software should be refactored, and develop a new scheme to precisely determine what refactorings should be applied to the identified places. Our approach is evaluated through an industrial study. The results show that our approach recommends refactorings more appropriately than a contemporary recommender.

Index Terms—requirements management; just-in-time requirements; traceability; refactoring;

I. INTRODUCTION

For many successful and long-lived software projects, the traditional notions of up-front requirements engineering (RE) — where requirements are formally specified and then baselined for subsequent development — offer limited value [1, 2]. Instead, requirements in these projects are captured less formally and become more elaborated only after the implementation begins. This is known as *just-in-time* RE [3, 4] which encompasses two key aspects: lightweight representation (defining requirements at a convenient level of detail) and evolutionary refinement (clarifying them when needed). Projects adopting such practice include both open source systems like Mozilla [4] and proprietary ones such as IBM’s Rational Team Concert [5].

Managing requirements in a just-in-time manner is not without risk. One challenge refers to the stagnation of the requirements that developers have already started to implement. A recent study investigated 157 user stories and showed that about 10% failed to be realized even after certain coding efforts were made [5]. Another major issue concerns the hindrance to the fulfillment of future requirements. Just-in-time RE, as currently practiced, lacks explicit “big picture” thinking [4]. As a result, a long-term penalty is often imposed on how well the software can be extended to accommodate new and changing requirements.

A mainstream method to facilitate future extensions and adaptations of a software system is refactoring [6, 7]. The basic idea of refactoring is to keep the internal software

quality under control during evolution by applying a set of behavior-preserving transformations to the code base. It is our overarching hypothesis that these transformations, if applied appropriately, can advance the state of the practice in software engineering by enabling more just-in-time requirements to be fully satisfied.

Testing this hypothesis entails addressing several specific research questions, including where, when, and how frequently to refactor, what refactoring(s) to perform, and how to assess the refactoring effect [8]. Murphy-Hill and Black [9] refer to the “when” and “how frequently” choices as refactoring tactics, and further suggest “refactoring early and refactoring often” is the principle that will benefit most developers. Contemporary approaches, however, depart from this principle by relying passively on developer’s spontaneity to trigger refactoring. Consequently, a number of refactorings may be missed or conducted later than expected [10]. The impacts are negative: Few refactorings can result in poor software quality and delayed refactorings can incur high maintenance cost.

In our earlier work, we devised an instant refactoring framework for actively assisting developers with better tactics [11]. The framework deploys a monitoring mechanism to analyze code changes constantly (e.g., between saves), and then prompts the developer for potential refactorings. The direct effect is that more refactorings are made without much delay [11]. We thus believe this framework sets up a firm and promising basis for handling those just-in-time requirements that are slowly progressing or hard to accommodate. While our work so far has focused on the “when” and “how frequently” tactics [11, 12], identifying “where” to apply “what” refactoring(s) remains one of the most pressing and vital issues confronting the research community [8].

In this paper, we tackle this pivotal issue via the novel use of requirements traceability. To locate more accurately “where” the code base should be refactored, we synthesize clustering-based link retrieval [13] with the as-needed traceability information captured in issue tracking repositories [4]. To determine more precisely “what” refactoring(s) should be applied to the identified places, we develop a new scheme by first examining requirements semantics as they relate to implementation, then leveraging the semantic characterization to uncover the problems in the code (i.e., bad smells [7]) that may impede the fulfillment of the requirements, and finally

choosing the type of refactorings that can remove the bad smells.

The contributions of our work lie in the strategic use of requirements traceability to change how code refactoring can be practiced to better fit its purpose — to make the software more extensible and adaptable to new requirements [8, 9]. In our opinion, the work represents an example of innovation *through* RE¹ in that an extensively researched requirements topic, namely traceability [14], creates principled ways to improve system development. In what follows, we present background information on just-in-time RE and software refactoring in Section II. We then detail our traceability-enabled refactoring framework in Section III. Section IV describes an industrial study to evaluate our approach. Finally, Section V places our work in the context of related research and Section VI concludes the paper.

II. BACKGROUND

A. Just-In-Time Requirements Analysis

In a recent attempt to understand RE in open source software projects, Alspaugh and Scacchi [2] identified three characteristics of what they termed “classical requirements”: 1) a central document (e.g., an IEEE-830 software requirements specification); 2) described in terms of problem space constructs (e.g., stakeholder goals); and 3) examination of the document to uncover defects (e.g., ambiguity, inconsistency, and incompleteness). This last point offers probably the greatest benefit, as asserted by Boehm with data support [15]:

Clearly, it pays off to invest effort in finding requirements errors early and correcting them in, say, 1 man-hour rather than waiting to find the error during operations and having to spend 100 man-hours correcting it.

Because of this benefit, the effort spent in getting the requirements right (e.g., modeling and formal methods) early in the software life cycle can be justified. Ernst and Murphy [4] referred it as “up-front RE”.

What was found out, however, is ongoing development with neither classical requirements [2] nor up-front RE [4]. This is true for a number of long-lived and critical open source systems that provide quality and rich functional capabilities, such as Apache and Mozilla [16].

What was actually practiced, then, is regarded as “just-in-time RE” where requirements are expressed less formally and only fully elaborated once the implementation begins [3, 4]. Contrary to up-front RE’s detailed plans and estimates for what should be done, just-in-time RE promotes rapid development and continuous refinement. The basic tenet is given succinctly by Beck [17]:

As we develop, we learn more about the problem and what is required to solve the problem.

¹RE’14 has a dual theme focused on innovation: innovation *in* RE and innovation *through* RE. See <http://re14.org> for more information.

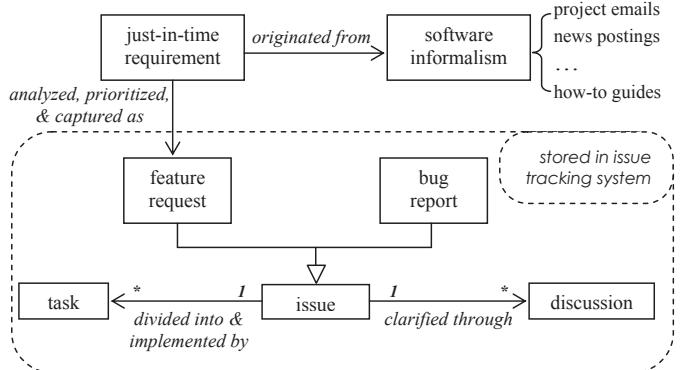


Fig. 1. Just-in-time requirements management via issue tracking system.

The common and dominant way to manage just-in-time requirements is the use of issue tracking system, e.g., Bugzilla² or JIRA³. We model the relevant concepts in Fig. 1 by synthesizing related studies in the literature [2, 4, 5, 16, 18]. The model is by no means a complete reference but is intended to serve as a unified conceptual basis for streamlining our discussion. Scacchi [16] recognized in his seminal work a set of “software informalisms” as the main sources for requirements. Informalisms, like emails and instant messaging, provide socially lightweight mechanisms for communicating and coordinating project knowledge.

Just-in-time requirements may be represented by user stories or in a feature list [4], but as shown in Fig. 1, the to-be-implemented ones “eventually end up as feature requests in an issue tracking system” [18]. Feature request and bug report are two special kinds of issue. For our discussion, they are different because “a defect (bug) stops living once resolved, but the description of a requirement (feature) is still valid documentation once implemented” [18]. The implementation of a feature typically involves a series of clarifying discussions. The actual implementation is then carried out by dividing the feature request into concrete and actionable tasks and by assigning the tasks to developers.

Because of the rich information stored in issue tracking systems, they are adopted by many projects besides open source systems to keep the requirements on track. For example, IBM’s Rational Team Concert project, which follows an iterative development process with six-week release cycles, relies on an issue tracking system for geographically distributed sites to clarify the requirements and to coordinate developers’ implementation tasks [5]. Studying the requirements clarification patterns in this project reveals a mixed outcome of just-in-time RE. While certain requirements progress smoothly toward full implementation, others are clarified late in the development cycle which makes them infeasible to be realized [5]. Two of the suspicious patterns are displayed in Fig. 2. They are representative of just-in-time RE’s drawbacks. Because requirements are defined at a convenient level of detail, erroneous and wasteful implementation can happen, as shown

²<http://www.bugzilla.org>

³<https://www.atlassian.com/software/jira>

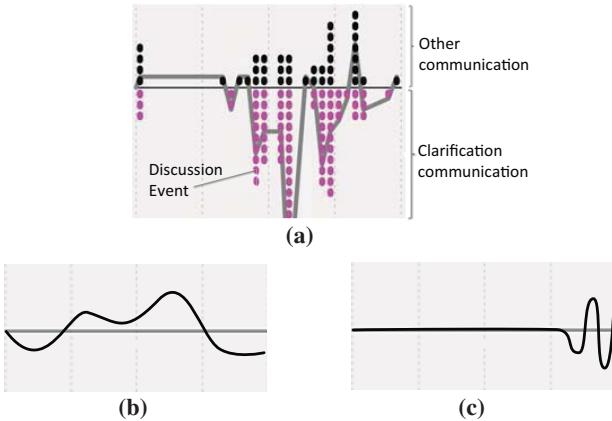


Fig. 2. Requirements clarification patterns where time in each graph proceeds left to right and depicts a release cycle. (a) Transforming discrete discussions (dots) into a pattern trajectory (line). (b) Back-to-draft. (c) Procrastination. All the graphs, (a)–(c), are adopted from [5].

by the “back-to-draft” pattern in Fig. 2b. Because requirements are clarified when needed, developers sometimes wait till the last minute to engineer them, as shown by “procrastination” in Fig. 2c. The procrastinated implementation is at greater risk of not being completed, but even if it is finished before the release, higher maintenance cost may be incurred [4].

In sum, being lightweight and flexible makes just-in-time RE a better fit for many software development projects than classical up-front RE, but often this comes at the cost of not fully meeting the project’s and stakeholders’ goals.

B. Software Refactoring

Although developers have long practiced program restructuring, the term “refactoring” was first introduced by Opdyke [6] as “the process of changing a software system in such a way that it does not alter the external behavior of the code, yet improves its internal structure”. The core idea is to redistribute programming elements (e.g., variables and methods) across the class hierarchy in order to facilitate future adaptations and extensions [8]. Clearly, the main objective of refactoring is to help adequately prepare the software system for meeting new and changing requirements.

Fowler [7] popularized refactoring when he cataloged 72 structural changes observed repeatedly in various programming languages and application domains. Research confirms that refactoring has become mainstream and is now widely practiced by developers [9, 10, 19]. A state-of-the-art survey by Mens and Tourwé [8] shows that the refactoring process consists of a number of distinct activities:

1. Identify where the software should be refactored.
2. Determine what refactoring(s) should be applied to the identified places.
3. Ensure that the applied refactoring preserves behavior.
4. Apply the refactoring.
5. Assess the effect of the refactoring.
6. Maintain the consistency between the refactored code and other artifacts such as requirements and tests.

Performing these activities requires different tools as the manual approach is often repetitive, time-consuming, and

error-prone — an example would be to apply RENAME IDENTIFIER refactoring across multiple class files. One area in which extensive tool support has been developed is “bad smell” detection. Bad smells are “structures in the code that suggest the possibility of refactoring” and 22 such signs of potential problems are initially identified in [7].⁴ A review of these 22 code smells suggests an imbalanced support in that **Duplicated Code** receives most research attention whereas some smells like **Message Chains** receive little [20]. The review also points out the lack of evidence available to justify the use of current smell-based approaches to direct effective refactoring [20]. Thus our knowledge about “where” to apply “what” refactoring(s) remains insufficient.

The list of activities given by Mens and Tourwé [8] is especially important from the viewpoint of developing automated tool support. Human intervention, however, is indispensable in software refactoring. A primary reason stems from the intrinsically informal interpretation of “behavior preservation”. Although Opdyke’s original definition hinges on formally checking refactoring preconditions and input-output behaviors [6], it is later proven to be insufficient [8]. Rather, it is recommended that we should have a wider range of definitions of behavior that need to be preserved by refactoring, depending on domain-specific or even user-specific concerns [8]. For example, the temporal constraints should be preserved when real-time software is refactored, and so should the concrete notions of safety (e.g., liveness) be preserved in the refactoring of mission-critical systems.

Another area that automated support is limited is the tactical dimension of refactoring choices. Such decisions include not only the previously mentioned “when” and “how frequently” to refactor [9], but also how to mix refactoring with developers’ other programming tasks (e.g., bug fixing). On one hand, refactoring can be conducted over protracted periods, during which developers perform few if any other kinds of program changes. For example, Microsoft typically reserves 20% of development efforts on thorough refactoring right after a version is released and before the development of the next version is initiated [21]. On the other hand, refactoring can be seen as a means to reach a specific end (e.g., adding a feature) instead of the end itself. For example, more than half of the Mylyn refactorings indicate that developers intersperse other kinds of program changes with refactorings to keep the code healthy [10]. These competing but often coexisting tactics make it difficult to tease out and measure the impact of refactoring.

In sum, refactoring is a mainstream software engineering practice whose prime purpose is to make the code base more extensible and easily adapted to future requirements. Despite the rapidly growing body of research in the last decade, there remains an important gap in the current knowledge about how to locate the “where” and determine the “what” for effective refactoring. The emerging research thrust in refactoring tactics

⁴More bad smells, together with refactorings that may help dispel them, can be found at <http://refactoring.com>.

further illuminates the challenge of assessing the value and effect of refactoring.

III. ENABLING SOFTWARE REFACTORING THROUGH REQUIREMENTS TRACEABILITY

The underlying premise of our research is that using just-in-time requirements to drive software refactoring will advance the state-of-the-art in both fields. Our essential argument is that the role of refactoring is better justified by leading more requirements to their full implementation. To examine this premise, we significantly extend monitor-based refactoring [11] by exploiting the novel use of requirements traceability information.

An overview of the proposed framework is presented in Fig. 3. As the developer performs coding and debugging activities, our framework monitors the code changes and relates the programming activities to requirements with the possible help of the issue tracking system. Through tracing and semantic characterization of the requirements, our framework recommends refactoring opportunities to the developer.

Compared to our prior work [11], the key difference which is also the unique advantage of the approach in Fig. 3 is *requirements-driven*. Unlike contemporary refactoring approaches that scan largely the source code to detect bad smells, our framework makes full and explicit use of the requirements information (i.e., feature requests extracted from the issue tracking system) to direct the refactoring process. In this way, refactoring is made to not just clean up the unhealthy code, but to do so with a purpose — to be better ready for satisfying more requirements. The rest of this section describes in detail how we leverage requirements traceability (Section III-A) and a new semantic scheme (Section III-B) to drive software refactoring. Section IV presents an evaluation of our approach.

A. Using Traceability to Locate “Where”

The specific question we address in this sub-section concerns the accurate identification of the places in the code that should be refactored. In our framework, we first consider the set of requirements that the current development cycle implements. This set can be readily extracted according to the issue tracking system’s “feature requests” (cf. Fig. 1), which we call “targeted requirements” with respect to the ongoing implementation. We then retrieve requirements-to-source-code traceability links for the targeted requirements, and detect code smells⁵ only within the tracing results. The simple rule that we are using here is that: “Do not refactor the code that cannot even be traced to the targeted requirements”. Thus the question of locating to-be-refactored code is cast as a requirements tracing problem.

Traceability is one of the most actively and intensively researched topics in RE [22, 23, 24, 25, 26]. A tracing method is *accurate* if it achieves a high level of recall and precision, where recall is defined as the percentage of correct links that are retrieved and precision is defined as the percentage of

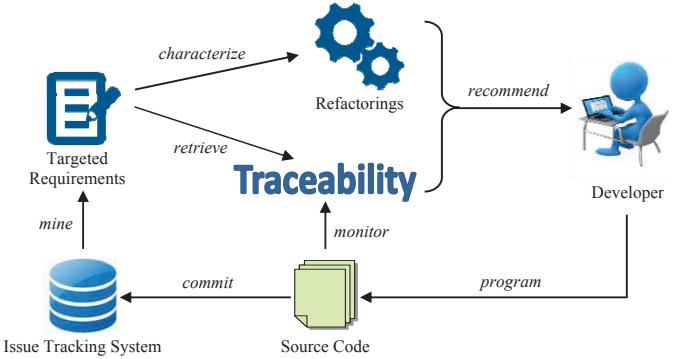


Fig. 3. Traceability-enabled refactoring framework.

retrieved links that are correct [27]. It is these accuracy criteria that our framework uses to guide the identification of “where” the software should be refactored. Although numerous tracing methods exist, the ones adopting standard information retrieval (IR) algorithms are equivalent in that they are able to capture almost the same traceability information [28]. In most cases, a recall of 90% is achievable at precision levels of 5-30% [24, 27, 28, 29].

Before choosing an accurate tracing mechanism among the many options [14, 30] or even developing a new one, we must understand how traceability is currently practiced in just-in-time RE. The pertinent literature, however, is surprisingly scant. A noticeable exception is Ernst and Murphy’s recent empirical study on three large open source software projects [4]. The tracking from requirements to implementation in these projects, perhaps not surprisingly, is labeled “as-needed traceability” [4]. Inheriting the essential characteristics of just-in-time RE, traceability is implemented where necessary, but as long as it *is* implemented, traceability is defined through the built-in support from the issue tracking system. All three projects follow the practice of prefacing a commit message to the code repository with a task ID number. In another word, the requirements-to-source-code traceability links can be obtained by querying the issue tracking system through the feature request’s task ID(s). The advantage of as-needed traceability is that developers actually do it [4]. It is therefore emphasized that automated traceability recovery is fruitless without taking into account the as-needed traceability information [4].

It should be pointed out that as-needed traceability does not equal to perfect traceability. For the requirements whose implementation is not yet finished, that is, those feature requests whose status is still “open”, the traceability information stored in the issue tracking system is incomplete at best. Even for the “closed” ones, if their implementation is procrastinated as illustrated in Fig. 2c, then the traceability information may be completely missing in the issue tracker due to the time pressure. Thus the requirements-to-source-code links defined by as-needed traceability are of high precision, i.e., they are indeed correct but are hardly complete.

Our framework integrates as-needed traceability into clustering-based link retrieval — a technique that we investigated in [13]. The rationale is derived directly from the

⁵The smell detection and refactoring recommendation in our framework are also requirements-driven and will be discussed in Section III-B.

TABLE I
GUIDING SMELL DETECTION AND REFACTORING SELECTION VIA REQUIREMENTS' SEMANTIC ANALYSIS

| Requirement's Action Theme | Issue ID* | Code Smell | Refactoring |
|----------------------------|----------------------------|-------------------------------------|--------------------------------------|
| Add | functional capability | Switch Statements | INTRODUCE LOCAL EXTENSION |
| | stakeholder role | Large Class Duplicated Code | EXTRACT CLASS |
| Enhance | quality attribute | Long Method | SUBSTITUTE ALGORITHM |
| Refine | expansion (more details) | Message Chains | ADD PARAMETER |
| | extension (more specifics) | Long Parameter List | EXTRACT SUBCLASS |
| Manage Dependency | abstraction | Duplicated Code | EXTRACT SUPERCLASS PULL UP METHOD |
| | invocation | Long Method Switch Statements | REPLACE METHOD WITH METHOD OBJECT |
| Remove | redundant feature | Duplicated Code Feature Envy | REMOVE CODE MOVE METHOD |
| Adapt | another operating system | Parallel Inheritance Hierarchies | EXTRACT INTERFACE |
| | another platform | Primitive Obsession | DYNAMIC METHOD DEFINITION |

* All the feature requests can be accessed from Mozilla's issue tracker at https://bugzilla.mozilla.org/show_bug.cgi?id=Issue_ID

```

1. For each req ∈ set of targeted requirements
2. VSM ← retrieve candidate traceability links based on the
   tf-idf textual similarity between code and req
3. C ← cluster VSM by applying single-link with k=8
4. Creq ← remove 3 lowest-quality clusters from C
5. If as-needed traceability ≠ NULL
6.   Creq ← remove those clusters containing no
      as-needed traceability from Creq
7. Return Creq

```

Fig. 4. Identifying to-be-refactored code by integrating as-needed traceability into clustering-based link retrieval.

cluster hypothesis stating that correct links tend to cluster near other correct links and farther away from incorrect ones. We examined five clustering algorithms (k -means, bisecting divisive, single-link, complete-link, and average-link) over four datasets, and the findings showed that the single-link algorithm at the 8-cluster ($k=8$) granularity best supports the cluster hypothesis in automated traceability [13]. The quality of the resulting clusters can then be judged on the basis of the link that is most similar to the requirement, and discarding the 3 lowest-quality clusters is found to significantly improve the tracing accuracy [13]. While this heuristic is kept intact, we introduce an additional filter so that if the as-needed (and correct) traceability links are defined, then the clusters that do not contain these links are further removed. The entire procedure is summarized in Fig. 4.

Note that clustering is only one way of feeding as-needed traceability into the retrieval process. Classification, for example, could lead to another automated solution [31]. Nevertheless, our underlying mechanism (lines 2–4 of Fig. 4) is shown to greatly enhance a baseline IR method [13]. The augmented filter (lines 5–6 of Fig. 4), as will be shown in Section IV, further improves tracing accuracy.

B. Characterizing Semantics to Determine “What”

Having identified the to-be-refactored code through traceability, we now turn our attention to refactoring selection. Current research chooses refactorings depending mainly on the bad smells directly observed in the source code. However,

bad smells are symptoms of design problems [7]. Different problems can lead to the same symptom, but their resolutions may differ. To handle a smell of **Duplicated Code**, for instance, we can simply REMOVE CODE to reduce redundancy or EXTRACT CLASS to achieve a better separation of concerns. A more extreme example is provided in Fowler's catalog: if a class is affected by **Type Code**, one can replace it in 5 different ways: CLASS, MODULE EXTENSION, POLYMORPHISM, STATE/STRATEGY, and SUBCLASSES [7]. Advancing the research on refactoring selection therefore requires a high *precision*, i.e., recommending the exact type of refactoring(s) by looking beyond the symptom (code smell).

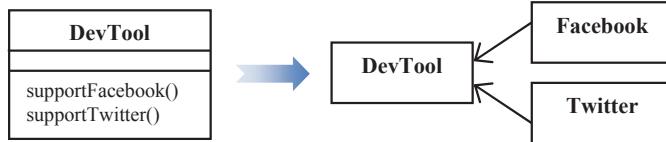
Our framework shown in Fig. 3 employs a *requirements-driven* component to direct smell detection and refactoring selection. In particular, we build upon the work on semantic analysis in RE [32, 33] to tease out the action-oriented theme of a requirement. Such theme, according to Fillmore's case theory [34], can be characterized by the *verb* in the requirements description and the *direct object* that the verb evokes (acts upon). We then detect those code smells that pose potential threats to the implementation of the intended action, and further recommend refactoring(s) to mitigate the threats. Table I shows the novel scheme, in which the requirements semantics helps unearth *why* the code smells so as to precisely determine *how* to correct the smell.

To guide the semantic classification, we extend the previous research [32, 33] and manually inspect the feature requests of Mozilla — a project that adopts just-in-time RE [4]. The analysis of Mozilla's issue tracking system was carried by two researchers independently. Their findings were then shared and discussed with the participation of a third researcher. Collectively, the most representative requirement was selected to report here. Next we ground our discussion on each of the semantic categories by using these representative requirements. Notably, the just-in-time requirements are stated heavily in solution-space terms, a deviation from “classical requirements” [2]. This, however, effectively facilitates our mapping between the requirement and its implementation.

1.1) Add a functional capability: Feature request 226680 is about providing a checkbox for installing extensions to profile directory so that the installation can be simplified. In complex procedures like installation, **Switch Statements** often appear but they violate object-orientation's encapsulation principle. To lead to the creation of a new functional capability, **INTRODUCE LOCAL EXTENSION** can be applied. Such a transformation is depicted as follows.

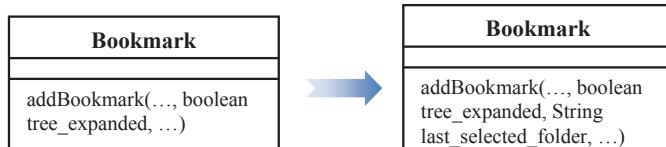


1.2) Add a stakeholder role: Issue 839959 requests new APIs (application programming interfaces) to be enabled for social media providers (e.g., Facebook, Twitter, etc.) to have direct access to Firefox's devtools. If **Large Class with Duplicated Code** exists in the code base, then one can **EXTRACT CLASS** to better modularize the individual stakeholder(s). This refactoring is illustrated below.



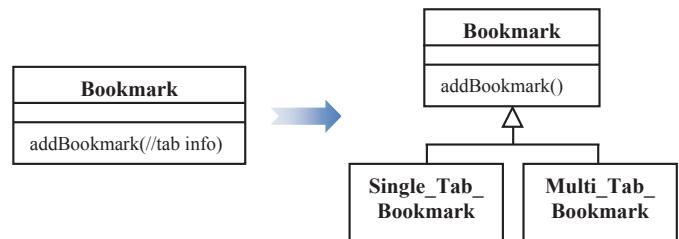
2) Enhance a quality attribute: For issue 578133, the heading reads, "make us fast". **Long Method** can cause not only performance issues but degradation of other quality attributes like complexity and understandability. **SUBSTITUTE ALGORITHM** helps replace the smell with the code that is faster, simpler, clearer, etc. Note that "make us fast" is truly a just-in-time requirement because its meaning becomes more unambiguous only after the implementation begins [4].

3.1) Refine by expanding the requirement with more details: Requirement 228585 builds on the bookmark-adding feature that remembers if the user last had the tree view expanded or not, and suggests a step further to remember the last selected folder in the tree view so that the folder can be highlighted. The intertwining relationship may cause **Message Chains** in the implementation, which can be addressed by **ADD PARAMETER** as follows.

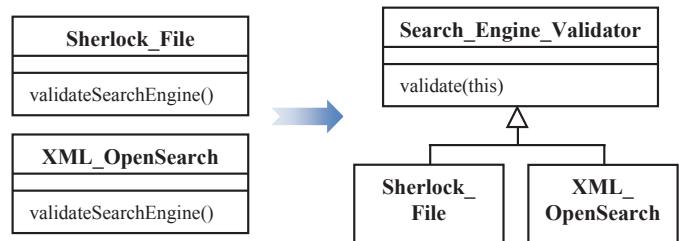


3.2) Refine by extending the requirement with more specifics: Feature request 713036 suggests extending bookmark operations from a single-tab to a multi-tab setting. **Long Parameter List** is a sign of potentially tangling code. The refactoring

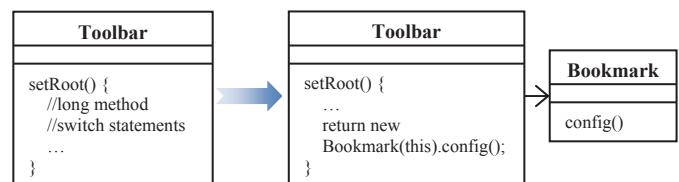
EXTRACT SUBCLASS delineated below can be used to refine the inheritance hierarchy.



4.1) Manage dependency via abstraction: Issue 341665 acknowledges the use of Sherlock file to validate search engine, but points out that there is no technical reason XML OpenSearch description could not be used. Validating search engine in different ways can result in scattered and **Duplicated Code**. **EXTRACT SUPERCLASS**, along with **PULL UP METHOD**, can then be performed to improve the code structure. This transformation is sketched as follows.



4.2) Manage dependency via invocation: Issue 239532 is raised because the user wants to allow the root of toolbar to be set with the bookmark folder. The tight interdependency between setting toolbar's root and various browser widgets can cause **Long Method** and **Switch Statements**. As shown below, **REPLACE METHOD WITH METHOD OBJECT** helps overcome the potential problems of these smells.



5) Remove a redundant feature: Issue 229482 requests to remove the status bar of bookmark as the displayed information is made available already in the manager window. Two code smells are relevant here. If `displayStatusBar()` of the `Bookmark` class is **Duplicated Code**, then **REMOVE CODE** can be triggered. If the method is too interested in another class (`Manager_Window`), it is a sign of **Feature Envy** and **MOVE METHOD** over `Manager_Window` will achieve a higher-cohesion and lower-coupling redistribution.

6.1) Adapt to another operating system (OS): Issue 251722 wants Mozilla running on Linux and Mac to offer the option

to import saved passwords same as Windows. Supporting multiple OSs can cause **Parallel Inheritance Hierarchies**, which may be refactored with EXTRACT INTERFACE.

6.2) Adapt to another platform: Feature request 669642 points out that about:firefox pops up the description in desktops, but in mobile platform, about:firefox should be better implemented by navigating to a window where one can check for updates and find help information. Accordingly, DYNAMIC METHOD DEFINITION may address the problem of **Primitive Obsession** (e.g., literals used to express the about:firefox message).

It is worth mentioning here that the categories listed in Table I are classified manually, though future automation is possible, e.g., by calibrating the Naive Bayesian classifier [5, 31]. Our analysis of Mozilla’s repository focuses on representative feature requests, though (semi-)automatically calculating the issue distribution over different categories can offer additional insight. Given the above reasons, the records presented in Table I may not be complete. In fact, at this stage of our research, we trade completeness for originality as our goal is to discover a new way to improve the precision of refactoring selection.

IV. EVALUATION

This section presents an empirical investigation into the impact of our traceability-enabled refactoring approach on just-in-time RE. While our overarching goal is to best transform and prepare the code to fulfill the requirements, the framework described in Section III is intended to achieve two specific objectives: (1) accurately locate the to-be-refactored code, and (2) precisely select the type of refactorings at these locations. Thus, we first introduce the background of our empirical study (Section IV-A), then assess separately how the two objectives are accomplished (Section IV-B and Section IV-C), and finally discuss the threats to validity of our study (Section IV-D).

A. Background

The subject system of our study is a software-intensive platform that provides technological solutions for service delivery and workforce development in a specific region of the United States. In order to honor confidentiality agreements, we use the pseudonym “WDS” to refer to the system. WDS has been deployed for over a decade and is developed using Java and Eclipse technologies. It intends to meet the goals of multiple stakeholders, including job seekers, employers, educational institutions, and government agencies.

WDS development follows an iterative and incremental life cycle with each version released in around 6 to 8 weeks. While WDS developers are encouraged to “refactor early and refactor often”, a designated refactoring period is reserved shortly after the latest version is released and right before the development of the next version is launched — a practice similar to Microsoft [21]. The requirements of WDS are managed in a typical just-in-time way. The project team uses the commercial state-of-the-practice JIRA issue tracking

system to communicate the requirements and to record the as-needed traceability information.

We select the most recent stable release of WDS to investigate. We refer to this version as “code base X”. This code base contains over 500 Java classes. The contextual information is shown in Fig. 5. The annotated t_a , t_b , and t_c in Fig. 5 should be thought of as short time periods (e.g., 1 or 2 days) rather than instantaneous time points. The just-in-time requirements extracted from JIRA for development cycle X include successfully closed feature requests $\{\text{REQ}_X_1, \text{REQ}_X_2, \dots, \text{REQ}_X_n\}$, as well as three open features $\{\text{REQ}_1, \text{REQ}_2, \text{REQ}_3\}$ that are not delivered at t_b . For the upcoming development cycle Y, two arriving requirements $\{\text{REQ}_4, \text{REQ}_5\}$ are already entered into JIRA at t_b , although more features $\{\text{REQ}_Y_1, \text{REQ}_Y_2, \dots, \text{REQ}_Y_m\}$ may be requested in the future.

For code base X at t_b , the set of targeted requirements is composed of $\{\text{REQ}_1, \text{REQ}_2, \text{REQ}_3, \text{REQ}_4, \text{REQ}_5\}$. It is this set that our framework takes full advantage of (cf. Fig. 3). The middle column of Table II provides the requirements description. As-needed traceability for $\{\text{REQ}_1, \text{REQ}_2, \text{REQ}_3\}$ can be found in JIRA, which shows partial implementation of these requirements. Further analysis of the discussion events indicates that REQ_1 and REQ_2 come “back-to-draft” (cf. Fig. 2b) whereas REQ_3 ’s implementation is procrastinated (cf. Fig. 2c). For this reason, REQ_1 and REQ_2 are marked as infeasible during development cycle X.

B. Assessing Accuracy of Code Location

As we formulate refactoring location as a requirements tracing problem, the accuracy is measured by standard IR metrics: recall and precision [27]. The answer set of all the correct traceability links is provided by the WDS project team at the requirement-to-Java-class granularity. Table III compares the automated tracing results performed by the tf-idf textual matching with uniform pruning [27], the clustering-based retrieval [13], and the clustering-based retrieval augmented with the additional filter (cf. Fig. 4).

The results in Table III show that, compared to the term-based tf-idf retrieval, grouping candidate traceability links into clusters and then filtering out the 3 lowest-quality clusters improve both recall and precision for all the 5 targeted requirements. The main reason is that correct links are pulling other correct links toward each other but pushing incorrect links away to different clusters [13]. Note that the clustering-based link retrieval works regardless of the availability of as-needed traceability, indicating the generalizability of our approach beyond just-in-time RE.

For $\{\text{REQ}_1, \text{REQ}_2, \text{REQ}_3\}$, having as-needed traceability serve as an extra filter further enhances the tracing performance. While recall is unchanged from the clustering-based retrieval, precision increases because the filter discards more false positives — those links appearing in the cluster that does not include any of the correct as-needed traceability links. The improvement, when compared with the precision of REQ_4 and REQ_5 , is nontrivial. Our results therefore stress

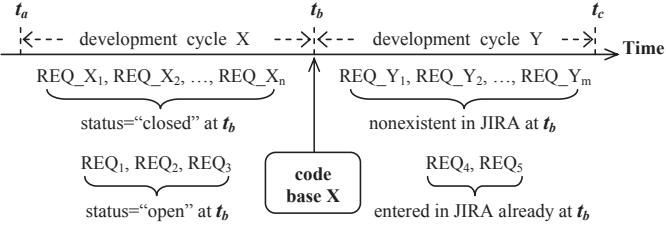


Fig. 5. Code base X (about 500 Java files) whose targeted requirements set at t_b is $\{REQ_1, REQ_2, REQ_3, REQ_4, REQ_5\}$.

TABLE II
DESCRIPTION AND CLASSIFICATION OF TARGETED REQUIREMENTS

| ID | Description | Semantic Category (cf. Table I) |
|------|--|------------------------------------|
| REQ1 | Provide a way to key youth achievements as part of student's academics | Refine by expansion (3.1) |
| REQ2 | Create the role of "WP Manager"† and allow it to run data integrity reports for individual staff | Add a stakeholder role (1.2) |
| REQ3 | Improve the merge utility to make it fast | Enhance a quality attribute (2) |
| REQ4 | Allow job-matching search on the public job order XML interface by accepting OSOC codes‡ | Refine by expansion (3.1) |
| REQ5 | Extend search facility from a single company to self-service participants | Refine by extension (3.2) |

† WP Manager (workforce program manager) is an emerging government role.

‡ OSOC (onetcenter.org) defines an occupation-specific national standard.

the importance of equipping automated traceability with the available as-needed information [4].

C. Assessing Precision of Refactoring Selection

We build on our tracing results to examine what refactoring(s) should be applied to the traced Java classes. Refactoring selection is essentially a recommendation problem [35]. In software engineering, a recommender helps to overcome developers' information overload by exposing them to the most interesting and relevant items — in our case, what code needs to be refactored in which way in order to correct what bad smells. Ultimately it is up to the developers to follow or ignore the recommendations.

Refactoring recommendations are often evaluated in two different ways. The first involves collecting opinions from domain experts, application developers, or other human evaluators. For example, Bavota *et al.* [36] recruited 30 Master's students in Computer Science to rate the usefulness of the suggested EXTRACT CLASS refactorings on a 5-point Likert scale. The second injects recommendations into actual development. For example, a prototype recommender, InsRefactor, was developed in our previous work and a controlled experiment was conducted to compare the students' refactoring performance with and without InsRefactor in order to quantify the recommendations' effectiveness [11].

We followed mainly the qualitative evaluation mechanism by asking the opinions from three WDS developers. In particular, we provided each developer with a hard copy of the targeted requirements (REQ₁–REQ₅) and asked them

TABLE III
ACCURACY OF LOCATING CODE FOR THE TARGETED REQUIREMENTS

| ID | <Recall, Precision> | | |
|------|-------------------------------|----------------------------|-------------------|
| | term-based retrieval (tf-idf) | clustering-based retrieval | additional filter |
| REQ1 | <.91, .08> | <1.00, .20> | <1.00, .44> |
| REQ2 | <.73, .31> | <.95, .38> | <.95, .46> |
| REQ3 | <.80, .34> | <.85, .39> | <.85, .61> |
| REQ4 | <.82, .11> | <.82, .33> | <.82, .33> |
| REQ5 | <.85, .09> | <.88, .12> | <.88, .12> |

to individually rate the "precision" (1: very inappropriate, 2: not appropriate, 3: neutral, 4: somewhat appropriate, 5: appropriate) and the "perceived implementation difficulty" (1: very difficult, 2: somewhat difficult, 3: neutral, 4: easy, 5: very easy) of the recommended refactorings with respect to the targeted requirements.

The refactorings that the WDS developers assessed included the recommendations generated by our approach and the ones made by InsRefactor [11]. In our study, InsRefactor worked by taking the traced Java classes as input and calling a set of state-of-the-art smell detectors to recommend refactorings.⁶ Although our approach used the same set of smell detectors, the refactorings were selected based on the scheme defined in Table I. This difference, though important for our internal data analysis, was *not* shown to the external evaluators (i.e., the WDS developers). In fact, all the recommendations, irrespective of the source, were displayed using the same style in Eclipse [11]. Basically the evaluator was prompted with the tuple \langle code, smell, refactoring \rangle in an arbitrary order to provide the ratings, and optionally, the justifications.

Using *requirements* to drive refactoring is the unique feature of our approach. As shown in Table II, each of the 5 targeted requirements fits in one and only one of the semantic categories. This demonstrates the generalizability of our scheme, and more importantly, leads to refactorings more on the target. For the 9 classes listed in Fig. 6a, our approach makes 6 recommendations shown in Fig. 6b. InsRefactor, meanwhile, recommends 11 refactorings, two of which overlap with ours. Although InsRefactor's recommendations were perceived easier to implement (average=3.7; average of ours=3.5)⁷, they were less appropriate (average=2.5; average of ours=3.8). Moreover, the refactorings targeted at previously infeasible requirements, namely REQ₁ and REQ₂, received positive ratings. One WDS developer commented, "Extract a WP Manager class is exactly what I'm doing right now [to implement REQ₂]." Not all of our recommendations received positive ratings. The one targeted at REQ₃ ("make the merge utility fast"), for instance, seemed appropriate but finding the actual algorithm to substitute for the long method was not an easy implementation task. The results thus show the precision of our refactoring selection and also indicate the room for improvement.

⁶A prototype implementation of InsRefactor as an Eclipse plug-in can be found at <http://www.sei.pku.edu.cn/~liuhui04/tools/InsRefactor.htm>.

⁷Cohen's kappa of pairwise ratings among the three WDS developers is 0.77 for "precision" and 0.63 for "perceived implementation difficulty"; both show substantial inter-rater agreements [37].

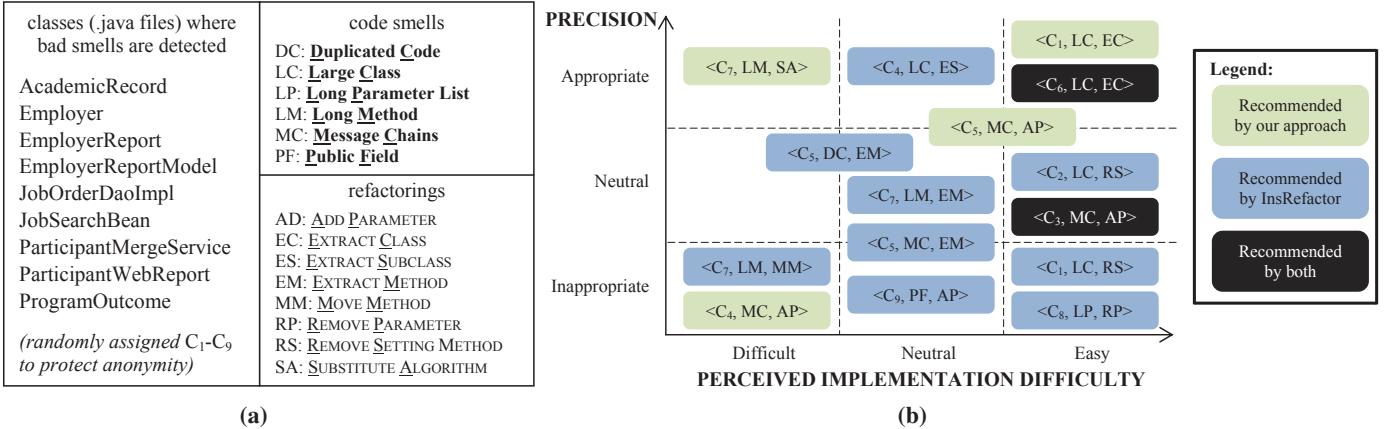


Fig. 6. (a) List of class files, code bad smells, and recommended refactorings. (b) Average ratings of the refactoring recommendations which are plotted in coarse-grained scales of negative, neutral, and positive along each axis; recommendations with a tied rating are placed arbitrarily within an area.

D. Threats to Validity

Our study represents an initial attempt to assess the impact of traceability-enabled refactoring on the overall just-in-time RE practice. We discuss here some of the most important factors that must be considered when interpreting the results.

The *construct validity* [38] of our study can be affected by two major threats. On the operational level, we measured appropriateness of recommended refactorings only qualitatively by surveying WDS developers. We were unable to evaluate the quantitative effect by interrupting the developers within their daily work, or to determine whether our approach would have a direct impact on the actual implementation (quality, speed, etc.) of the just-in-time requirements. On the conceptual level, our guiding premise is that the more requirements implemented the better. While this may not be desired for processes like up-front RE, we believe it fits well with the lightweight nature of just-in-time RE which lacks “big picture” thinking [4]. One may think it is careless planning to even start implementing the features that are redundant or of little interest to the stakeholders, but we argue that oftentimes we do not know what features become irrelevant and removable until they are delivered.

One of the *internal validity* [38] threats relates to the sequential examination of the two steps in our approach, i.e., locating the “where” followed by selecting the “what”. While this order is in line with the 6 refactoring activities defined by Mens and Tourwé [8] (cf. Section II-B), caution must be taken in interpreting our findings as a whole rather than separately. In particular, because the automated tracing results did not reach the 100% recall and precision (cf. Table III), it is not known how a perfectly traced set of code artifacts would affect the refactoring recommendations and their ratings. It is also not known how incorrect and imprecise traceability may have affected our current results. Another confounding variable stems from our design of basing the subjective rating on the targeted requirements. While all the three WDS developers in our study provided justifications directly related to REQ₁-REQ₅, carrying out an experiment with actual requirements implementation tasks can mitigate this threat to a great extent.

An important threat to *external validity* [38] is that our study was conducted on a single dataset. For requirements-to-source-code tracing, our approach relies upon the clustering-based link retrieval which is shown to perform accurately in several, but not all, project environments [13]. For refactoring recommendation, it is encouraging to note that our semantic categorization is in general applicable to the open source Mozilla and the proprietary WDS projects. We therefore position our study as a proof-of-concept stepping stone toward more rigorous analyses and assessments.

V. RELATED WORK

Software traceability research has its root in Gotel and Finkelstein’s seminal work [22]. A majority of the effort has been spent in *recovering* the traceability links [30]. Our work advances the literature by showing that traceability can be more than recovered. It can be *utilized*. Research in this direction includes using traceability to support software change tasks [39] and examining trace granularity (requirement-to-class vs. requirement-to-method) from a cost-benefit perspective [25]. Our work focuses on a different application area: code refactoring. Interestingly, we support the finding that class-level traces are more practically valuable [25] because our approach can recommend refactorings at both class and method levels (cf. Table I).

Utilizing traceability does not have to wait till the end of the project. In [40, 41], Cleland-Huang *et al.* proposed event-based traceability to generate timely project notifications by monitoring changes made to software artifacts. We also built a monitor to promptly remind the developer of resolving the code smells at their early appearance [11]. Our current framework thus institutes a “monitor” in Fig. 3 to allow for instant refactoring recommendations. We plan to combine the requirements-driven and monitor-based refactoring mechanisms in the next release of InsRefactor.

Tahvildari and Kontogiannis [42] were among the first to use requirements to guide software re-engineering. They encoded softgoal interdependency graphs to reason about code transformation’s (partial) satisfaction of the quality goals like performance and maintainability. Our work complements theirs by

leveraging both functional and quality requirements to guide code transformation. A recent endeavor explored how different types of refactoring might help or hurt traceability [43]. In a word, our research differs from and also helps close the loop of that endeavor: The work in [43] is about “refactoring for traceability” and ours is about “traceability for refactoring”.

VI. CONCLUSIONS

In this paper, we have contributed a novel traceability-enabled refactoring approach to addressing the challenges of just-in-time RE. Our approach takes full advantage of the requirements traceability information to identify the to-be-refactored locations and to rationalize why a certain type of refactoring should be carried out. A proof-of-concept study on an industrial software system shows our approach’s accuracy of locating “where” to refactor and precision of recommending “what” to refactor.

Our overarching goal is to improve traceability and refactoring practices by fully exploiting their interconnections. Our future work includes enriching the requirements-driven scheme for refactoring, pushing the relevant recommendations instantly to the stakeholders, and maintaining traceability after refactorings are performed.

ACKNOWLEDGEMENT

We are grateful to the partner company for the generous support of our research and Zhangji Chen for assisting in data analysis. The work is in part supported by the U.S. NSF (National Science Foundation) Grant CCF-1238336. Authors from Beijing Institute of Technology are supported by the National Natural Science Foundation of China (No. 61272169), Program for New Century Excellent Talents in University (No. NCET-13-0041), and Beijing Higher Education Young Elite Teacher Project (No. YETP1183).

REFERENCES

- [1] J. Aranda, S. Easterbrook, and G. Wilson, “Requirements in the wild: how small companies do it,” in *RE*, 2007, pp. 39–48.
- [2] T. A. Alspaugh and W. Scacchi, “Ongoing software development without classical requirements,” in *RE*, 2013, pp. 165–174.
- [3] M. Lee, “Just-in-time requirements analysis — the engine that drives the planning game,” Agile Alliance, Technical Report, May 2002.
- [4] N. A. Ernst and G. C. Murphy, “Case studies in just-in-time requirements analysis,” in *EmpIRE*, 2012, pp. 25–32.
- [5] E. Knauss, D. Damian, G. Poo-Caamaño, and J. Cleland-Huang, “Detecting and classifying patterns of requirements clarifications,” in *RE*, 2012, pp. 251–260.
- [6] W. F. Opdyke, “Refactoring object-oriented frameworks,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1992.
- [7] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.
- [8] T. Mens and T. Tourwé, “A survey of software refactoring,” *IEEE TSE*, vol. 30, no. 2, pp. 126–139, February 2004.
- [9] E. Murphy-Hill and A. P. Black, “Refactoring tools: fitness for purpose,” *IEEE Software*, vol. 25, no. 5, pp. 38–44, Sept/Oct 2008.
- [10] E. Murphy-Hill, C. Parnin, and A. P. Black, “How we refactor, and how we know it,” *IEEE TSE*, vol. 38, no. 1, pp. 5–18, January/February 2012.
- [11] H. Liu, X. Guo, and W. Shao, “Monitor-based instant software refactoring,” *IEEE TSE*, vol. 39, no. 8, pp. 1112–1126, August 2013.
- [12] H. Liu, Y. Gao, and Z. Niu, “An initial study on refactoring tactics,” in *COMPSAC*, 2012, pp. 213–218.
- [13] N. Niu and A. Mahmoud, “Enhancing candidate link generation for requirements tracing: the cluster hypothesis revisited,” in *RE*, 2012, pp. 81–90.
- [14] S. Nair, J. L. de la Vara, and S. Sen, “A review of traceability research at the requirements engineering conference,” in *RE*, 2013, pp. 222–229.
- [15] B. Boehm, “Software engineering,” *IEEE Transactions on Computers*, vol. 25, no. 12, pp. 1226–1241, December 1976.
- [16] W. Scacchi, “Understanding the requirements for developing open source software systems,” *IEE Software*, vol. 149, no. 1, pp. 24–39, February 2002.
- [17] K. Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.
- [18] P. Heck and A. Zaidman, “An analysis of requirements evolution in open source projects: recommendations for issue trackers,” in *IWPSE*, 2013, pp. 43–52.
- [19] M. Kim, T. Zimmermann, and N. Nagappan, “A field study of refactoring challenges and benefits,” in *FSE*, 2012, article no. 50.
- [20] M. Zhang, T. Hall, and N. Baddoo, “Code bad smells: a review of current knowledge,” *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 23, no. 3, pp. 179–202, April 2011.
- [21] M. Cusumano and R. Selby, *Microsoft Secrets*. Free Press, 1998.
- [22] O. Gotel and A. Finkelstein, “An analysis of the requirements traceability problem,” in *ICRE*, 1994, pp. 94–101.
- [23] K. Pohl, “PRO-ART: enabling requirements pre-traceability,” in *ICRE*, 1996, pp. 76–85.
- [24] J. H. Hayes, A. Dekhtyar, and J. Osborne, “Improving requirements tracing via information retrieval,” in *RE*, 2003, pp. 138–147.
- [25] A. Egyed, F. Graf, and P. Grünbacher, “Effort and quality of recovering requirements-to-code traces: two exploratory experiments,” in *RE*, 2010, pp. 221–230.
- [26] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, J. I. Maletic, and P. Mäder, “The quest for ubiquity: a roadmap for software and systems traceability research,” in *RE*, 2012, pp. 71–80.
- [27] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, “Advancing candidate link generation for requirements tracing: the study of methods,” *IEEE TSE*, vol. 32, no. 1, pp. 4–19, January 2006.
- [28] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia, “On the equivalence of information retrieval methods for automated traceability link recovery,” in *ICPC*, 2010, pp. 68–71.
- [29] A. Mahmoud, N. Niu, and S. Xu, “A semantic relatedness approach for traceability link recovery,” in *ICPC*, 2012, pp. 183–192.
- [30] M. Borg, P. Runeson, and A. Ardö, “Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability,” *Empirical Software Engineering*, (accepted).
- [31] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, “Automated classification of non-functional requirements,” *Requirements Engineering*, vol. 12, no. 2, pp. 103–120, April 2007.
- [32] S. Liaskos, A. Lapouchnian, Y. Yu, E. S. K. Yu, and J. Mylopoulos, “On goal-based variability acquisition and analysis,” in *RE*, 2006, pp. 76–85.
- [33] N. Niu and S. Easterbrook, “Extracting and modeling product line functional requirements,” in *RE*, 2008, pp. 155–164.
- [34] C. Fillmore, “The case for case,” in *Universals in Linguistic Theory*, E. Bach and R. Harms, Eds. New York: Holt, Rinehart and Winston, 1968, pp. 1–88.
- [35] M. P. Robillard, R. J. Walker, and T. Zimmermann, “Recommendation systems for software engineering,” *IEEE Software*, vol. 27, no. 4, pp. 80–86, July/August 2010.
- [36] G. Bavota, A. De Lucia, A. Marcus, and R. Oliveto, “Automating extract class refactoring: an improved method and its evaluation,” *Empirical Software Engineering*, (accepted).
- [37] J. L. Fleiss, B. Levin, and M. C. Paik, *Statistical Methods for Rates and Proportions*. Wiley-Interscience, 2003.
- [38] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.
- [39] P. Mäder and A. Egyed, “Assessing the effect of requirements traceability for software maintenance,” in *ICSM*, 2012, pp. 171–180.
- [40] J. Cleland-Huang, C. K. Chang, and M. J. Christensen, “Event-based traceability for managing evolutionary changes,” *IEEE TSE*, vol. 29, no. 9, pp. 796–810, September 2003.
- [41] J. Cleland-Huang, P. Mäder, M. Mirakhori, and S. Amornborvornwong, “Breaking the big-bang practice of traceability: pushing timely trace recommendations to project stakeholders,” in *RE*, 2012, pp. 231–240.
- [42] L. Tahvildari and K. Kontogiannis, “A software transformation framework for quality-driven object-oriented re-engineering,” in *ICSM*, 2002, pp. 596–605.
- [43] A. Mahmoud and N. Niu, “Supporting requirements traceability through refactoring,” in *RE*, 2013, pp. 32–41.

Supporting Traceability through Affinity Mining

Vincenzo Gervasi^{*†}, Didar Zowghi[†]

^{*}Dipartimento di Informatica, Università di Pisa, Italy

[†]School of Software, University of Technology Sydney, Australia

Abstract—Traceability among requirements artifacts (and beyond, in certain cases all the way to actual implementation) has long been identified as a critical challenge in industrial practice.

Manually establishing and maintaining such traces is a high-skill, labour-intensive job. It is often the case that the ideal person for the job also has other, highly critical tasks to take care of, so offering semi-automated support for the management of traces is an effective way of improving the efficiency of the whole development process.

In this paper, we present a technique to exploit the information contained in previously defined traces, in order to facilitate the creation and ongoing maintenance of traces, as the requirements evolve. A case study on a reference dataset is employed to measure the effectiveness of the technique, compared to other proposals from the literature.

I. INTRODUCTION

In a talk at RE’2003 about measuring the mismatches between areas of interest of researchers and practitioners in RE [1], Martin Feather lamented *requirements traceability* as a practically important yet under-served area of RE research. In the 10 years since, his call has been taken up by several groups of researchers, and numerous techniques to help provide automatic or semi-automatic support for the establishment and maintenance of traces between requirements artifacts have emerged.

Early proposals have drawn mostly from the repertoire of Information Retrieval (IR) techniques (e.g., using Vector Space Model and TF-IDF approaches), essentially positing that *lexical* similarity between two documents (or fragments thereof, e.g. single requirements) is a strong indicator that the two should be linked in a traceability relation. These approaches, while useful and reasonably effective as an initial attempt, suffered from certain limitations, namely:

- 1) They ignore the different nature of different relationships between requirements. A trace can embody a technical refinement, a goal-means relationship, a make-break relationship, a requirement-test, etc., but the only criterion considered is the occurrence of the same terms in the two artifacts.
- 2) They ignore that artifacts of different nature can employ different sublanguages — for example, marketing terms on one side, technical jargon on the other. In such circumstances, relying on lexical similarity will yield unsatisfactory results.
- 3) They do not leverage the effort already put into establishing links. In effect, every new artifact is treated as a fresh new query into an IR system, with the results

used as candidate for establishing traces, and with no memory of the past.

To overcome these limitations, more recent approaches have applied Machine Learning (ML) techniques instead of or in addition to classical IR techniques. In general terms, ML approaches seek to extract information from pre-existing traces, use the information to build a model of the previous linking patterns, and leverage the model in suggesting candidates for future traces. Such techniques help overcome limitations 1) and 3) mentioned above, in that the future suggestions tend to mimic as far as possible what already had been done manually in the past, thus leading to an efficient re-use of the effort and knowledge put into establishing an initial set of traces.

As an aside, it should be noted that the pre-existing traces are not necessarily the “perfect” ones. It is entirely possible that an automated technique can identify relevant traces that had been missed by the human analyst. However, given the current state of the art, human analysts tend to be significantly more effective at establishing traces than existing algorithms, when traceability is considered at all. Indeed, in common industrial practice – and especially in small-to-medium enterprises – the real problem seems to be having any traceability in place at all, rather than having a perfect set of traces.

In this work, we focus on limitation 2) above, i.e. on those cases where different terms, choice of wording, style and level of abstraction are used in diverse, but related, requirements documents that need to be linked. Typical examples include linking marketing requirements to technical specifications, or business goals to requirements and to use cases or scenarios. At the same time, we apply the general framework of ML approaches, thus addressing 1) and 3) as well. Results show that even simple techniques can be remarkably effective in assisting a human analyst in the task of establishing traces.

Our main contribution in this work is a simple technique based on the concept of *affinity*, which is a measure of how the occurrence of certain pairs of terms increases the likelihood that two requirement artifacts should be linked, based on historical data. We formally define the problem of incremental traces construction, propose a solution based on affinity, and validate the solution by means of two experiments on industrial datasets, comparing our results with others published in the literature.

The paper is structured as follows. In Section II we lay out our problem in formal terms, and establish needed notation. Section III describes the technique we are proposing for identifying candidate links, based on affinity mining; this is followed by an account of our experimental validation in Section IV. We discuss the applicability of the technique, and

possible extensions, in Section V. This is followed by a review of related work, and some conclusions.

II. PROBLEM FORMALIZATION

We consider a scenario in which two sets of textual artifacts, each consisting of a single unit which we will call without loss of generality¹ a *requirement*, are to be linked with each other by means of *traces* of unspecified (but homogenous) semantics. A trace between two requirements means that the two of them are related according to the considered semantics.

Due to historic usage, we will call the two sets of requirements H (for: high-level) and L (for: low-level), with $H = \{h_1, \dots, h_n\}$ and $L = \{l_1, \dots, l_m\}$. The traces $E = \{e_1, \dots, e_p\}$, where each $e_k = (h_{i_k}, l_{j_k})$ are thus the edges of a bipartite graph $G = (H \cup L, E)$.

The problem we want to address can then be formalized as follows: given a current (non-empty) set of traces E , and a new incoming requirement h' , identify a set of requirements $L' \subseteq L$ such that the links (h', l') , where $l' \in L'$, represent the same relationship between h' and elements of L as the pre-existing links in E did for elements of H and elements of L .

A few observations should be noted. First, the problem is stated here in terms of a new incoming h' , but its symmetrical version with a new l' is totally analogous. We prefer the formulation with an incoming high-level requirement because this corresponds to a frequently occurring situation in practice, i.e. when a new user-level requirement is presented, and the analyst wants to check which technical requirements are related to the novel user request.

Second, the problem is stated in terms of the *insertion* of a new requirement in H , but *deletions* are easily modeled (by dropping from E all the links from or to the removed requirement), and *edits* can be modeled as a deletion followed by an insertion². We will discuss in Section V how such an extension could be exploited to support traceability maintenance.

Third, in this paper we consider the simplest case with traces between just two sets of artifacts, and simple edges between two requirements at a time. In certain scenarios, we could find multiple artifacts or hyperlinks instead (i.e., links between a subset of H and a subset of L , for example when a single marketing requirement is *implemented-by* a set of technical requirements, but not by any one of them alone), or different set of links, with different semantics (i.e., E is partitioned into subsets E_1, \dots, E_z , for example when certain edges represent *implemented-by* whereas others represent *conflicts-with*). For expository purposes, we will keep to the simplest case in illustrating the technique, and will discuss later in Section V how our approach could be extended to these more complex scenarios.

¹I.e., we are not interested in the present work in their particular role as goals, marketing requirements, technical requirements, bug reports, etc.

²As a minor technical detail, in a real system the knowledge extracted from links that are dropped following a deletion should be preserved, whereas knowledge extracted from links that are explicitly deleted should be discarded, since in that case it is the trace itself that has been deemed invalid.

III. PROPOSED TECHNIQUE

In this section we will first present the intuition behind our notion of *affinity* and the rationale for its application in supporting the establishment of traces between requirement artifacts. We will then provide a technical description of the operations to be performed during the *training* phase (when affinity values are mined from pre-existing traces) and during the *application* phase (when mined affinity values are used to suggest candidate links for a new incoming requirement).

A. Rationale

In most practical cases, the information that is carried by the establishment of a link between two requirements expresses the fact that the linked requirements predicates over the same or related domain objects, or concern identical or related actions to be performed, or they specify that the same event has to be detected to trigger some action, etc. Such relationships are often hinted at by *lexical* elements in the requirements: e.g., when two requirements talk about the same system component, identified by a specific name, they are probably related.

Approximating the desired relationship by means of lexical similarity has been found to be an effective technique in many real-world applications (see, among others, the study on linking marketing requirements to technical requirements in [2], [3]). The underlying theory is that if the same terms appear in each of the two requirements, they stand a good chance of being related. From that theory, numerous methods based on classical Information Retrieval techniques (such as TF-IDF [4]) have been proposed, validated via experiments, and ultimately implemented in tools.

Those techniques, however, are applicable only when the exact same terms are used in both H and L . In many cases, the language used in the two artifacts tend to differ, e.g. H could be users' requests for new features in the software (expressed in terms of the users' domain vocabulary), whereas L could be technical specifications of the internal structure of the software (expressed according to the developers' vocabulary).

In our earlier work [5], we advocated using *affinity* between terms from H and L (i.e., a measure of how frequently requirements in which certain pair of terms occurred, were linked in pre-existing data) instead of mere occurrence of the same terms, in order to estimate the likelihood that a given pair of requirements (one of them new) should be linked with each other.

More precisely, in [5] we studied how affinity data mined from pre-existing requirements links could provide information about domain glossaries. Among the findings, we demonstrated on a case study that the technique found that almost 90% of the terms used in requirements had a positive affinity towards themselves. In other words, affinity agrees with techniques based on occurrences of the same terms on 90% of the cases. However, in the remaining 10% of the cases, the occurrences of identical terms provided a *negative* contribution to the likelihood of links between requirements. For those terms, techniques such as TF-IDF would lead to a wrong candidate being suggested. Additionally, affinity mining identified a

r = “The DPU-TMALI shall configure the ping-pong frame limit at startup as specified by TMALI_PP_LIMIT provided during initialization. The default value shall be M frames and shall be capable of being modified dynamically.”

$$T(r) = \{ \text{TMALI_PP_LIMIT, VALU, M, LIMIT, DPU-TMAL, DEFAULT, MODIFI, FRAME, SPECIFI, BE, DYNAM, CONFIGUR, CAPABL, STARTUP, PROVID, INITI, PING-PONG } \}$$

Fig. 1. An example for the term-extraction function T .

large number of pairs of different terms that had a positive contribution to the linking likelihood, and that would be ignored by techniques based on occurrences of identical terms. Thus, on an artificial subset of the requirements from a publicly available case study, selected such that the probability of a link between two requirements would be on average 50%, our affinity-based technique exhibited a remarkable 95.7% recall and precision, compared to 86.5% for the standard cosine correlation measure.

B. A Measure for Affinity

In the present work, we used a simplified version of affinity, computed from pre-existing links as follows.

Each of the two requirements documents, H and L , is pre-processed, and for each requirement r (either an h_i or a l_i) a set of terms $T(r) = \{t_1, \dots, t_{p_r}\}$ is extracted by means of a standard POS-tagger³. We consider the stemmed form of nouns, adjectives, adverbs and verbs as terms, ignoring instead prepositions, conjunctions, disjunctions, pronouns, interjections, particles, determiners, modal verbs, numerals, and various punctuation. An example for T , applied to a real requirement from the case study described in Section IV, is shown in Figure 1.

Since we consider *sets* of terms, information about the number of occurrences is lost at this stage. More refined measures of affinity might potentially use such information. However, in our tests requirements were rather terse, so that rarely a term occurred more than once in a requirement. As a consequence, ignoring multiple occurrences does not have a significant impact for the kind of documents we are considering.

Once T is available, affinity α between terms is computed as follows:

$$\forall t_h \in \bigcup_{h \in H} T(h), t_l \in \bigcup_{l \in L} T(l)$$

$$\alpha(t_h, t_l) = \#\{(h', l') \in E \mid t_h \in T(h') \wedge t_l \in T(l')\}$$

In intuitive terms, $\alpha(t_h, t_l)$ is a count of how many links already exist between high-level requirements in which t_h occur, and low-level requirements in which t_l occurs.

It should be noted that the measure α is a rather crude approximation of our concept of affinity, since we simply count the number of links already established between requirements

³As will be better described in Section IV, we used the OpenNLP suite for linguistic processing.

that include the given pair of terms. This definition, however, has the advantage of being easily understandable and intuitively sound, and particularly easy to apply in practice even where there is no sophisticated tool available for requirements traceability⁴.

Another interesting observation is that α need not be scaled according to document size, since its definition is only significant in the context of two given requirement documents (H and L), and hence any normalization to try to make it independent from the the size of H and L would not provide any real advantage.

C. Learning

Armed with the definitions above, the process of learning affinity from pre-existing data can be simply stated. Given two requirements documents H and L , and a set of already established traces E , compute α for all terms appearing in $H \cup L$, according to its definition.

The learning phase needs not be repeated from scratch every time a change occurs in H , L or E . In fact, our definition for α supports easy differential updating without requiring massive recomputations. For example, if a requirement h is removed from H , causing the cascade removal of a certain sets of links $E' \subseteq E$, where $e = (t_h, t_l) \in E' \implies t_h \in T(h)$, it is sufficient to subtract 1 to the affinity scores of all pairs $\alpha(t_h, t_l)$. In the same way, the establishment of a new link corresponds to a simple increment of the corresponding $\alpha(t_h, t_l)$.

Such differential updates can be done in real-time. This is important, because the technique enables taking into account linking decisions by the human analyst immediately, e.g. when clicking to confirm a link candidate proposed by a requirements management tool, the sets of further candidates can be updated accordingly to incorporate the additional information provided by expert confirmation.

D. Application

The application scenario that we consider in this paper is having a tool to support a human analyst in taking linking decisions when a new requirement is considered for inclusion in a requirements document.

Let $G = (H \cup L, E)$ be the pre-existing set of traces, T be the term-extracting function based on linguistic processing as described above, and $\alpha_{G,T}(\cdot, \cdot)$ be the corresponding affinity measure.

When a new requirement, say h' , is presented for inclusion in H , the analyst has to identify which pre-existing requirements in L should be linked with h' . Our proposal provides a ranked list of requirements that are candidate for linking, with the highest-likelihood candidates (according to affinity score) at top.

⁴In fact, in [5] a more sophisticated measure was defined, which included term frequency and inverse document frequency as part of the definition.

In detail, the rank of each $l \in L$ in the list of candidates for being linked to h' is determined according to its score $\rho_{h'}(l)$ defined as

$$\rho_{h'}(l) = \sum_{\substack{t_{h'} \in T(h') \\ t_l \in T(l)}} \alpha(t_{h'}, t_l)$$

The human analyst can then examine the ranked list, and confirm or reject linking suggestions according to his or her own judgement.

For presentation purposes, the list of ranked candidates can be limited to a certain threshold for $\rho_{h'}(l)$, say τ , or to a fixed number of elements, say k . Alternatively, the list could be returned in its entirety, and then the analyst could be instructed to stop perusing the list after a certain number of consecutive items are judged not suitable for linking. The most appropriate choice for limiting the list returned can vary based on the particular context of application, so we do not investigate here the effect of different policies (in the next section, we will use a variable threshold τ for measurement purposes).

IV. EXPERIMENTAL EVALUATION

We provide an evaluation based on an experiment, with a comparison to two other techniques proposed in the literature for the same problem.

A. Method

Our main hypothesis is that *affinity*, as approximated by our measure α , is a good predictor for whether a new requirement should be linked to a set of pre-existing ones. The alternate (null) hypothesis is that affinity is no better than random chance in determining whether two requirements should be linked.

Additionally, we are interested in measuring how good our affinity-based method fares compared to two other methods: Vector Space model (weighted according to TF-IDF) [4], and Reinforcement Learning [6].

Our reference for a correct set of links will be the judgement of an experienced analyst, familiar with the problem. To remove any threat of researcher bias, we use a dataset (both H and L requirements, and a set of manually-established traces E) that were developed independently from the current research endeavour. To facilitate comparison with other techniques, we use a dataset that has been developed in an industrial context and that has been used by other researchers in other traceability-related studies.

All computations are performed by computer programs implementing the various formal definitions from Sections II and III, thus reducing the threat of clerical errors in processing the data.

Being based on a single experiment, our method could refute our main hypothesis in case of failure, but in case of success cannot prove the generalizability of our proposed technique to different datasets or contexts. Thus, results can serve as a first indication of effectiveness, but a larger, systematic study would be needed to ultimately prove its wider applicability.

B. Experimental Design

1) *Dataset*: We used a published dataset for CM-1, a scientific instrument (to be carried onboard a satellite) whose requirements were developed by NASA and made available to the scientific community as part of the NASA Metrics Data program.

The full dataset consists of numerous documents, including defect reports with links to code artifacts. For our study, we considered two textual documents: a Requirements document (our H) consisting of 235 requirements, mostly 1–3 paragraphs long, and a Design document (our L) consisting of 220 design description statements. A total of 361 links were manually established between these two documents. The full dataset has a very low link density: of 51700 possible pairs (h, l) , only 361 (0.7%) were correct links.

For the purpose of comparing to the Reinforcement Learning method from [7], we measured the performances of our Affinity method on a subset of the full CM-1 requirements, the same subset used in [7] and there named CM1SUB. This reduced dataset consists of 22 high-level requirements (H) and 53 low-level design statements (L), linked by a total of 45 links (E). CM1SUB has a slightly higher link density (3.9%) compared to CM-1, yet the task of automatically identifying the 45 correct links among the 1166 possible pairs remains challenging.

2) *Linguistic Processing*: We used the OpenNLP toolset [8] for the needed preprocessing at the linguistic level. OpenNLP can be configured for specific applications by training it on a given corpus; however in our experiment we wanted to simulate the situation where no special knowledge about the requirements language, style or domain is needed beforehand. Hence, we used the standard models for English distributed with OpenNLP itself for *sentence detection* (used, for example, to distinguish when a dot is just part of an abbreviation, or when it indicates a full stop ending a sentence) and *tokenization* (used to identify lexically significant chunks of text inside a sentence). After sentence detection and tokenization, a requirement r (either from H or L) consisted of a sequence of sentences, each consisting in a sequence of lexical tokens.

Part-of-speech tagging was performed on each sentence by using OpenNLP's maximum entropy POS tagger, which classifies tokens according to a slightly enriched variation of Penn's Treebank set [9]. As we described in Section III-B, only tokens classified as nouns, adjective, adverbs and verbs were considered; tokens in other classes were ignored and discarded at this stage. Moreover, the filtered sequence of tokens was flattened into a set at this stage, eliminating duplicates.

As a last step, stemming was performed by applying the popular Porter's stemmer [10] to the set of tokens, thus completing the calculation of $T(r)$. See Figure 1 for an example of the results of the whole process.

3) *Building the Affinity Model*: The affinity model (i.e., the values for $\alpha(\cdot, \cdot)$) is obtained by a straightforward implementation of the definition. In particular, all pre-existing traces are considered in sequence. For each trace $e = (h, l) \in E$, the affinity value between all terms in $T(h)$ and all terms in $T(l)$ is incremented by 1 (starting at 0).

It may be observed that while a simple increment by 1 may seem simplistic, in practice what we obtain is a count of how many links exist in the given data set between requirements containing at least one occurrence of the given terms. Since the actual values of $\alpha(\cdot, \cdot)$ are only used for comparison purposes, i.e. in ranking the list of candidates, their value is immaterial, and any more complicated function that still preserves the ordering would have no net effect on the results. On the other hand, we would be very wary of any function that does not preserve the ordering, i.e. one that would rank a pair of terms appearing in fewer links as more tightly related than another pair that appear in more links. Thus, a principle of economy leads us to stick with the simpler formulation.

4) *Identifying Candidate Links:* Once $\alpha(\cdot, \cdot)$ is mined from the data, identifying candidates for linking is simply a matter of computing the cumulative affinity score for two requirements, the new one h^* (to be inserted in H), and the pre-existing ones $l \in L$.

Again, the process involves a straightforward implementation of the definition of $\rho_{h^*}(\cdot)$. This is obtained by first computing $T(h^*)$, then summing, for each l , the affinity measure of each pair of terms from $T(h^*)$ and $T(l)$.

In this case, too, we could perform a normalization step; however since values of $\rho_{h^*}(l)$ are only used to rank different l s, and only in the context of the same h^* , normalization is not needed.

In order to keep the sets of candidates manageable by the analyst, we used a parametric threshold τ : any pair (h^*, l) with $\rho_{h^*}(l) \leq \tau$ would not be included in the list of candidates. All pairs above the threshold would be included, with an understanding that pairs with higher ρ values would be presented in a more prominent role (e.g., at the top of a list) to the user for validation.

5) *Measuring Results:* In order to measure the quality of the candidate links, we resorted to the classical Information Retrieval measures of *precision* and *recall*, computed in a k -fold validation scheme. More precisely, we performed 22 measures; on each measure, a different requirement h^* was removed from H , and all the links (h^*, l) , with $l \in L$, were removed from E .

The system was then trained on the remaining data, constituting a reduced dataset G^* , and the corresponding $\alpha_{G^*, T}(\cdot, \cdot)$ measure was computed for all remaining pairs. We then simulated the arrival of the “new” requirement h^* , computed the ranked list of suggested candidate links according to $\rho_{h^*}(\cdot)$, and measured precision and recall of that list compared to the correct set of links (those that were removed from E). The results were then averaged over the 22 instances.

In some practical cases, the quality of the topmost (i.e., highest ranking) candidates would be more relevant than those of the candidates further down in the list. This happens, for example, when the semantics of links is something like *duplicate-of*; simply finding that a user request (or bug report) is a duplicate of a pre-existing (and already accepted) feature request, is sufficient to decide whether the new requirement should be considered for inclusion or not. In other cases, though,

the aim is to find *all* significant links, e.g. when the semantics is, say, *contributes-to-goal* or *blocks-implementation-of*.

In our experiments, we adhered to the second (and more taxing) model, i.e. the aim was to find all relevant links, not just one or a few of the “best” ones. Accordingly, we measured precision and recall comparing the full set of candidates (truncated based on a threshold τ) to the full set of manually established links for each requirement h^* . Moreover, to study how the choice of τ affects results, we considered different values for τ , evenly spaced in 20 steps between the minimum and maximum value of ρ returned for a given h^* .

Let us call $E_\tau^c(h^*)$ the list of candidates (h^*, l) (with $\rho_{h^*}(l) > \tau$) returned by our method, and $E^g(h^*)$ the set of manually-established links in the original E (mnemonic: superscript c for *candidate*, g for *gold set*).

For each value of τ , we considered a *true positive* any link that was suggested as a candidate by our technique and was also manually established in the original data, and a *true negative* any link that was neither suggested as a candidate by our technique, nor present in the manually-established links. Conversely, a link suggested by our technique but not manually established in the original data would be considered a *false positive*, and a link that was manually established, but not returned as a candidate, would be a *false negative*.

These four sets can be defined formally as follows:

$$\begin{aligned} TP &= E_\tau^c(h^*) \cap E^g(h^*) \\ TN &= \{(h^*, l) \mid l \in L \wedge (h^*, l) \notin E_\tau^c(h^*) \cup E^g(h^*)\} \\ FP &= \{(h^*, l) \mid l \in L \wedge (h^*, l) \in E_\tau^c(h^*) \setminus E^g(h^*)\} \\ FN &= \{(h^*, l) \mid l \in L \wedge (h^*, l) \in E^g(h^*) \setminus E_\tau^c(h^*)\} \end{aligned}$$

Precision and recall (at threshold τ) are thus defined as usual; moreover since we are essentially conducting a binary test, we can also measure *accuracy*, e.g. how many of our classifications (true positives or true negatives) are correct, as a proportion of all possible tests. Another useful measure that combines precision and recall by computing their harmonic mean is the *F-score*⁵; this can be useful as a synthetic indicator of overall effectiveness of a classification method.

Formally, these measures are given by

$$\begin{aligned} Prec &= \frac{\#TP}{\#TP + \#FP} \\ Rec &= \frac{\#TP}{\#TP + \#FN} \\ Acc &= \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN} \\ F &= 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec} \end{aligned}$$

C. Results

We can finally present the results from our experiment on the CM1SUB dataset, providing a comparison with other

⁵The F-score can be tailored to assign different weight to either precision or recall. Here we used the balanced F-score, also called *F1*, that weights them equally.

TABLE I
RESULTS FROM THE EXPERIMENT ON THE CM1SUB DATASET, AT VARYING THRESHOLDS.

| τ | Prec | Rec | Acc | F | #P |
|--------|-------|-------|-------|------|------|
| 0% | 0.051 | 1.000 | 0.055 | 0.10 | 40.0 |
| 5% | 0.078 | 1.000 | 0.396 | 0.14 | 26.3 |
| 10% | 0.108 | 0.956 | 0.594 | 0.19 | 18.2 |
| 15% | 0.174 | 0.911 | 0.775 | 0.29 | 10.7 |
| 20% | 0.255 | 0.867 | 0.864 | 0.39 | 7.0 |
| 25% | 0.358 | 0.867 | 0.914 | 0.51 | 5.0 |
| 30% | 0.515 | 0.778 | 0.951 | 0.62 | 3.1 |
| 35% | 0.611 | 0.733 | 0.963 | 0.67 | 2.5 |
| 40% | 0.727 | 0.711 | 0.972 | 0.72 | 2.0 |
| 45% | 0.853 | 0.644 | 0.976 | 0.73 | 1.6 |
| 50% | 0.880 | 0.489 | 0.971 | 0.63 | 1.1 |
| 55% | 0.882 | 0.333 | 0.964 | 0.48 | 0.8 |
| 60% | 0.857 | 0.267 | 0.960 | 0.41 | 0.6 |
| 65% | 1.000 | 0.178 | 0.958 | 0.30 | 0.4 |
| 70% | 1.000 | 0.089 | 0.954 | 0.16 | 0.2 |
| 75% | 1.000 | 0.067 | 0.952 | 0.13 | 0.1 |
| 80% | 1.000 | 0.044 | 0.951 | 0.09 | 0.1 |
| 85% | 1.000 | 0.044 | 0.951 | 0.09 | 0.1 |
| 90% | 1.000 | 0.044 | 0.951 | 0.09 | 0.1 |
| 95% | 1.000 | 0.022 | 0.950 | 0.04 | 0.0 |
| 100% | 1.000 | 0.022 | 0.950 | 0.04 | 0.0 |

approaches, and on the full CM-1 dataset, which more faithfully represent a larger traceability context

1) *CMISUB and Comparison*: Table I presents the values for our four measures at varying settings of τ (which, we recall, is expressed in percentiles over the range of ρ , and measured at 5% intervals).

The results are quite satisfying, with accuracy peaking at $\tau = 45\%$ on a value of 0.976 (that is: 97.6% of all classification decisions are correct). On a highly skewed data set as our CM1SUB, accuracy alone does not provide a full assessment: in fact, a trivial classifier that would simply return no candidates at all for any new incoming requirement, always answering “no” so to say, would *still* be right 96.1% of the times on CM1SUB (and a full 99.7% of the times on the entire CM-1 dataset!). However, our affinity-based technique also exhibits an F score of 0.73 at $\tau = 45\%$, with 85% precision and 64% recall, whereas the trivial classifier mentioned above would return no true positives, and hence would have 0% precision and recall.

Another positive feature that can be observed in Table I is that precision and recall are not overly sensitive to the exact value of τ , and thus the exact value of the parameter is not critical. Any value of τ between 40% and 60% would yield approximately 75%-85% precision, and any value between 20% and 40% would yield approximately 70%-85% recall. As usual, increasing precision values tend to decrease recall, and vice versa.

To give an idea of the cognitive burden placed on the analyst that has to browse the list of candidate links, in order to judge which ones are worthy of establishing, the last column in Table I reports the total number of positives returned ($\#P = \#TP + \#FP$), i.e. the size of the list of candidates presented on average for each new incoming requirement.

At $\tau = 0\%$, there is of course very little selection, and the technique identifies 40 of 53 low-level requirements as candidates for linking. Needless to say, presenting 75% of all possible link targets is not particularly useful: it might be easier to just consider all 53 low-level requirements, and at least rely on some guarantee of completeness.

But already at $\tau = 15\%$, only around 10 requirements are presented, which on average include over 90% of the real links. Roughly speaking, since on average each high-level requirement in CM1SUB is manually linked to just above 2 low-level requirements, and we have a recall of 0.91, this means that the analyst will have to distinguish among the 10 candidates between 2 real links, to be confirmed, and 8 false positives, to be discarded. Only in less than 10% of the cases the list of 10 candidates will not include a relevant link. For non-critical applications, this seems a good compromise between effort spent in confirming link candidates, and level of completeness of the recovered links: the technique saves 75% of the effort, at the cost of a risk of not identifying 10% of the links that would be established (at four times the cost) in the case of a complete analysis of all low-level requirements for each new incoming high-level requirement.

The exact way precision and recall are related can be visualised by plotting them on a precision-recall graph (see Figure 2), which we also use to compare the effectiveness of our affinity-based proposal to TF-IDF and Reinforcement Learning (RL).

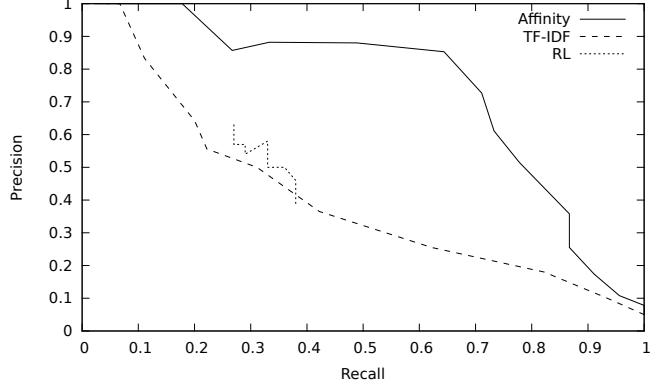


Fig. 2. Precision-Recall curve for Affinity, compared to TF-IDF and Reinforcement Learning, CM1SUB dataset.

The Affinity graph shows how precision is essentially stable for our technique for a large range of recall values (from 0.3 to 0.65), with a optimal spot around recall 65% and precision 0.85%. This corresponds to the optimal $\tau = 45\%$ that we had already identified in Table I.

More interestingly, it appears that Affinity substantially outperforms both TF-IDF and Reinforcement Learning⁶.

While reinforcement learning is still a relatively novel approach in traceability research, and thus its effectiveness is

⁶For the latter two methods, numerical data was obtained from [11]; the resulting diagram is essentially the same as the one shown in Figure 6 in [7].

not yet well established, it is somewhat surprising that Affinity performs so much better than TF-IDF. However, we should consider that Affinity implicitly identifies and incorporates the relationship between occurrences of identical terms that is at the heart of TF-IDF, while discarding those terms that present no evidence of being related in pre-established traces, despite occurring in multiple requirements, and adding those terms that are not identical, but frequently occur in linked pairs of requirements.

For example, Affinity identifies that high-level requirements containing the word **BUFFER** are often linked to low-level requirements containing **BUFFER**, just as TF-IDF would suggest. But in addition, it also identifies that high-level requirements containing **COMMAND** are even more frequently (actually, two times more frequently) linked to low-level requirements containing **TASK**, an association that TF-IDF would be unable to make. In a similar vein, Affinity is able to mine from pre-existing data that **ERROR** is often related to **ENQUEU** (the stem for *to enqueue*), which derives from the specific way errors are handled in the CM-1 system, and could not be derived by purely lexical means without looking at pre-established traces.

2) *CM-1 and Scalability*: The CM-1 dataset, with its larger size and lower link density, poses additional difficulties for an automated selection of candidate traces.

CM-1 has six times more requirements compared to CM1SUB. What is worse, since the number of potential links grows quadratically with the number of requirements, there are 44 times more potential links, and only 8 times more correct links. It is thus to be expected that the performance of any technique aiming to support traceability would drop considerably on larger datasets. Indeed, as Figure 3 shows, Affinity on CM-1 performs worse than on CM1SUB: but not dramatically worse. The optimal compromise here is slightly lower: the best value for F is found at $\tau = 50\%$ (pleasantly close to the optimum for CM1SUB, which was found at $\tau = 45\%$), with 77% precision and 63% recall.

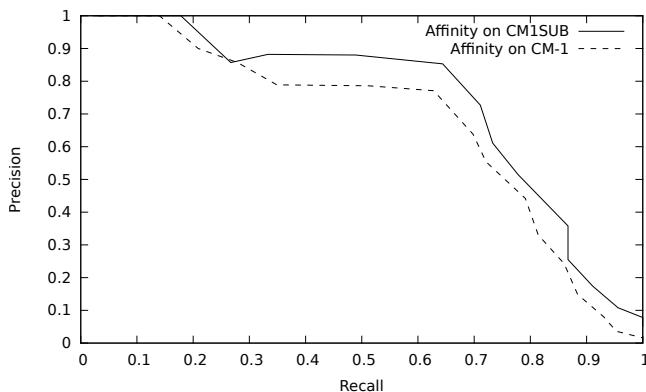


Fig. 3. Precision-Recall curve for Affinity on two different datasets: CM1SUB and CM-1.

Apparently these values are comparable to the 85% precision and 64% recall we had found for CM1SUB. But we have

to remember that these numbers represent slightly lower percentages of a much larger base value, and that the amount of effort that can be put in selection does not scale linearly (or, in other words: the human analyst has a limited capacity for attention and time that can be spent in examining trace candidates). As an example, for CM1SUB we hypothesized that 90% recall could be a target (i.e., the risk of missing 10% of the links was acceptable), and that lead to 17% precision, meaning that on average, the analyst had to find the 2 true links amidst 10 candidates. For CM-1, the same level of recall would lead to a precision of 8%, half what we had for CM1SUB. In other terms, the analyst has to find 1 true link amidst 10 candidates (a task that could be deemed twice as difficult to get right).

In most contexts, scalability would thus be an issue: while the theoretical performances of the approach are still good, and the computational cost is negligible (see Table II), in practice the attention span of the analyst can be a limiting factor.

TABLE II
COMPUTATION TIMES FOR VARIOUS STAGES OF THE AFFINITY TECHNIQUE,
ON A COMMODITY PC.

| Operation | Time (secs) |
|-------------------------------------|-------------|
| Initializing OpenNLP components | 2.7 |
| Loading and stemming specifications | 5.5 |
| Building affinity model | 2.1 |
| Testing and computing measures | 0.6 |

V. DISCUSSION AND APPLICABILITY

A. Generalizability and Tuning

The experimental results reported above support our main hypothesis, that Affinity can be an effective technique to identify candidate links in an ongoing requirements traceability process.

Of course, it would be quite daring to generalize from those experiments, only two, limited in size, and in the context of the same project, to all possible practical scenarios. The CM-1 dataset in fact exhibits certain peculiar properties (e.g.: highly stylized language, abundance of acronyms, rigorous naming conventions, high degree of textual polish of the requirements) that are all but universal in requirements documents. Hence, we do not claim generalizability beyond what the experiments have demonstrated.

Still, we see these results as a clear indication that relaxing the hypothesis that artefacts to link are homogeneous in language and vocabulary, can not only not be detrimental, but even improve the effectiveness of certain automatic techniques. Classical IR techniques developed for a scenario where queries were expressed in the same language as the documents to be retrieved, do not transition gracefully to software development scenarios, where links span vastly different categories of artefacts (e.g., from test cases to code). Even among requirements documents, authors and purpose can be so different that linked elements have often little lexical content in common.

The affinity concept can be applied in different ways, and extended to accommodate different needs. For example, links

representing different semantic relationships could be treated separately, and a special affinity score computed for each category. This view also accommodates bidirectional linking (e.g., traces between multiple versions of the same requirement in time, with forward traces meaning *evolved-into* and backward traces meaning *originates-from*).

Similarly, the exact way a requirement artefact is mapped to a set of terms is largely parametric. In our experiments, the mapping function T was based on standard linguistic processing. But if one were to apply affinity to program code, T could be implemented as a function to transform a unit of source code such as a function, class, or module, into a set of tokens (say: names of local variable, methods, etc.). The basic idea of mining previous knowledge encoded in pre-existing traces would still be applicable.

A number of different parameters and tools could be tuned, including the details of the linguistic processing steps (our T), the exact operational definition of a measure for affinity between terms (our α), or the way affinity between terms is combined to establish a ranking between candidate links (our ρ). In the present work, we have chosen the most basic options for each of these, in order to avoid complicating the presentation. However, in experimenting with alternative options (e.g.: using non-linear combinations for ρ instead of simple summation, or considering multiplicity in the set returned by T instead of a flat set of terms, or only considering affinity for values of α above a certain threshold, or adding a penalty for identical terms that were *not* linked, etc.), we have observed that the technique tends to be remarkably stable with respect to such changes. As we have already observed, the deciding factor is the relative ranking of candidate links, and the value of the threshold τ . The first is preserved by most reasonable alternatives, and the second was found experimentally to be not particularly critical in a large range of values. We thus anticipate that our results would survive small changes to the parameters or source data.

B. Other Application Scenarios

For expository purposes, we have framed our reference problem (Section II) in terms of an incoming new high-level requirement that has to be linked to pre-existing low-level requirements. This, however, is just one of the many potential application scenarios. In a more general view, affinity mining can be seen as a tool to be employed, among others, in the wider traceability maintenance picture.

In maintaining a set of traces, the various sets of artifacts (and the traces themselves) are assumed to evolve independently; the challenge is then to keep the traces up-to-date with the changes, spending as little human effort as possible. In this more complex setting, the knowledge extracted from established (and confirmed) traces needs not be discarded when the artifacts they used to link evolve. Once a confirmed link has established evidence that, say, “COMMAND” has high affinity to “TASK” (see Section IV-C1), this affinity score might well survive the removal of a requirement about one such command. Hence, in traceability maintainance the knowledge about affinity has to be maintained independently from its sources; the approach

would be more focused towards glossary construction and maintenance, with the affinity glossary being an independent repository, rather than a derived property of the pre-existing traces themselves.

We have not explored the maintenance setting in this work, choosing instead to focus on the simpler case with a stream of incoming new requirements. The basic case also lends itself well to investigation of application scenarios where affinity data is used, interactively, to support the writing of new requirements. For example, we could envision a scenario where upon arrival of a new high-level requirement, after checking that it is not a duplicate or already implemented, a corresponding new low-level requirement has to be written. An editing tool could then subtly suggest (e.g., as part of a word-completion functionality) those low-level terms that are already known to have high affinity with terms appearing in the text of the new high-level requirement.

In general, affinity could be used as a substitute or additional measure of relatedness between artifacts in a variety of contexts, beside the main one we discussed in this paper. We plan to investigate its effectiveness in those different roles as part of our future work in this area.

C. Threats to Validity

We have designed our experiments with a view to eliminate or mitigate most threats to validity; yet there are a number of threats that we could not eliminate.

Among the internal validity threats that we tried to address we can cite selection bias (the data set was selected based on its availability and previous usage by other researchers, and not based on the peculiar needs of our technique) and time-related effects such as memory, maturation, and repeated testing. In our design, the only human intervention happened well before the technique was developed, namely when the reference set of traces was manually defined by a human expert, and there could be no influence by the researchers on the original authors of the data set.

External validity is more problematic, since we only have experimental results from a single project, in two different data sets. Further experiences, and possibly replication studies, are needed before any claim of generalizability can be made.

There is also a threat about content validity, in that our measures (precision, recall, etc.) may not be able to capture those features that are really relevant for the analyst. Other factors (e.g., ease of use of an implemented tool) may play an important role, in that an analyst may be unwilling to spend much time working with an awkward tool, or be ineffective at link selection because of some user interface problem. However, those measures that we have used have been proven historically to correlate well with user’s effectiveness, and moreover provide a convenient way to compare different techniques (each of them, a different treatment) all else being equal. In this sense, our comparison to TF-IDF and Reinforcement Learning in Section IV provides solid evidence that Affinity is, among the three treatments we considered, the one providing the best performances on the same data.

For a single experiment, there is no issue of statistical validity; hence we have not endeavored to perform statistical significance tests on our results. In a future study, measuring performance differences on a significant number of different datasets, it may be needed to consider statistical validity before claiming generalizability.

VI. RELATED WORK

Among the vast scientific literature related to requirements tracing, we consider first those works that purpose to generate a set of candidate links for unlinked requirements.

Many early works advocated a direct application of classical Information Retrieval techniques, such as Vector Space Model (VSM) weighted by text frequency - inverse document frequency (TF-IDF) (among others, [12], [13], [2]). These techniques assumed to start with a clean slate, that is, from artifacts that were not linked at all; the problem was then to generate an initial set of links for the whole set of documents. Since no information other than the documents themselves was available, only lexical similarity was used in generating candidate links, and their applicability was restricted to artifacts that used the same vocabulary. In contrast, our proposal mines pre-existing links for context-dependent information, that is then used to relax the restriction that all artifacts should use the same vocabulary.

The limitations stemming from sole reliance on lexical content was quite rapidly felt. A successive stream of research tried to use Latent Semantic Analysis (in a static context as in [14] or in an evolutionary perspective as in [15]), or explicit dictionaries, thesauri or glossaries prepared for traceability purposes by a domain expert; the value of such a document was highlighted by several researchers as an important supplement to VSM-based techniques, e.g. [13]. In a sense, our approach is closer to this second strain, except that the vocabulary (relating the language used in one artifact, to a potentially different one used in another artifact) is inferred from pre-existing traces, rather than prepared by a domain expert. A similar idea has been recently advocated in [16], where an automated thesaurus builder is used in support to link generation.

Another idea for improving over the basic VSM model that has been proposed concerns using the location of terms, either to define a sort of *local neighbourhood* [17], or to assume *n*-grams as the unit of lexical reference, and sliding windows as the unit of relevance for linking purposes [18]. We have not addressed location of terms in our proposal: in fact, we explicitly discard any positional information in the computation of T . While positional information could potentially be useful, not just in terms of location, but also in terms of syntactic roles of the various terms, we have not considered the issue in this work. On the other hand, *n*-grams could be easily (and maybe more effectively) modeled by using even a basic nominal grammar in T , for example to assemble noun phrases, instead of considering each term on its own. We intend to investigate the issue in future studies.

The limitation of purely lexical approaches have been identified by several researchers. As a consequence, in more

recent times a systematic exploration of the applicability of Machine Learning techniques to the requirements traceability problem has been initiated.

The various ML approaches are so diverse that we cannot conduct a full survey here; we will simply mention a few of the most relevant ones. Already in 2004, ML was advocated as an alternative to lexical based techniques [19]. More recently, various ML techniques such as swarm intelligence [20] and reinforcement learning [7] have been applied; these are the research efforts that are closest in spirit to our own, and the ones we used for a comparison in Section IV. Reinforcement learning in particular was applied first to recover traces from normative codes (i.e.: standards, laws, regulations) to requirements in [21], where the technique was found to be particularly effective for the specific context. This might be in part due to the stable nature of one of the two sets of artifacts (the regulations), which are supposed to be the same over long periods of time and across different projects, thus providing good opportunities for reinforcement learning. In contrast, our affinity based approach is completely symmetric, and it is not expected that variability will occur on one class of artifacts only.

While all of the ML approaches assume either a given “golden” dataset to train a classifier of sort, or a given “oracle” to serve as a utility function (e.g., in approaches based on genetic algorithms, such as [22]), not all of them consider the ongoing evolution of multiple sets of artifacts as a source for traceability information, as our approach does. Traceability in an evolving context is thus the second major area of research which we briefly survey.

An adaptation of static IR techniques to the evolving context is presented in [23] (and more extensively in [24]), where different strategies are introduced and evaluated on a set of cases. A more thorough treatment, focusing on modeling the problems involved in trace evolution in addition to proposing a technical solution, is given in [25] in the context of traceability between requirements and design elements. Interestingly, [25] explicitly disclaims applicability to the initial construction of a set of traces, suggesting instead to use IR and data mining techniques for that. In contrast, we propose using IR and data mining techniques as part of ongoing evolution, and assume that the *bootstrap* is given by manually-established links (or, by traditional lexically-based IR techniques).

A fuller view of traceability activities as parts of a larger process is provided in [26]. In fact, our affinity-based approach could be configured as a part of a more complex evolution workflow, and possibly integrated with other techniques. Indeed, a recent trend is emerging towards composing traceability environments out of distinct components, and optimizing the various parameters for the task at hand. In its most sophisticated incarnation, customization of tools and parameters is obtained by applying genetic algorithms to a configuration, and letting the population of settings evolve until a reasonable optimum is obtained [27]. Affinity mining could then be one of the techniques and measures used in a rich toolbox of complementary approaches.

VII. CONCLUSIONS

We have presented a technique to exploit the information contained in previously defined traces among a set of software requirements, in order to facilitate the creation and ongoing maintenance of the traces as the requirements evolve, new requirements are defined, or old ones are edited or deleted.

The results from our experiments show that the proposed technique, based on a measure of *affinity* between pair of terms that embodies the likelihood that the terms will appear in requirements that are linked with each other, is substantially more effective than TF-IDF, in particular in cases where the artifacts to be linked employ different languages, jargon, or domain vocabularies. This is a frequently occurring scenario in many software development processes, e.g. when there is a need to link users' requests for new features (expressed in the users' jargon) to technical requirements (as written by analysts and developers) that implement them.

We also compared the effectiveness of our approach with a ML technique based on Reinforcement Learning, again finding substantial improvement.

While no claim of generalizability can be made at this stage, we believe that further studies will draw a large scope for affinity-based techniques, in specific contexts.

This work has revolved around proving the viability of the basic technique. As part of future work we intend to investigate the improvements that can be obtained by more refined linguistic processing, as well as testing the technique on a wider variety of scenarios.

Our validation has been entirely technical so far. This is both an advantage (in that the experiments are fully replicable, and there is no human variability involved), and a disadvantage (in that we have not considered human factors that might impact the practical usability of the technique). Hence, an avenue for further improvement is certainly deploying our technique in the context of a live industrial project, in order to gather first-hand users' feedback.

REFERENCES

- [1] M. S. Feather, T. Menzies, and J. R. Connolly, "Relating practitioner needs to research activities," in *Proc. of the 11th IEEE Int. Requirements Engineering Conf.* Los Alamitos, CA: IEEE CS Press, Sep. 2003, p. 352.
- [2] J. Natt och Dag, V. Gervasi, S. Brinkkemper, and B. Regnell, "Speeding up requirements management in a product software company: Linking customer wishes to product requirements through linguistic engineering," in *Proc. of the 12th IEEE International Requirements Engineering Conference*, Kyoto, Japan, Sep. 2004.
- [3] —, "A linguistic engineering approach to large-scale requirements management," *IEEE Software*, vol. 22, no. 1, Jan./Feb. 2005.
- [4] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [5] V. Gervasi and D. Zowghi, "Mining requirements links," in *Proceedings of the 17th International Conference on Requirements Engineering: Foundation for Software Quality*. Essen, Germany: Springer-Verlag, 2011.
- [6] A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [7] H. Sultanov and J. H. Hayes, "Application of reinforcement learning to requirements engineering: requirements tracing," in *Proc. of the 21st IEEE Int. Requirements Engineering Conf.*, Jul. 2013, pp. 52–61.
- [8] Apache Foundation, "Apache OpenNLP," <https://opennlp.apache.org/>, Oct. 2013.
- [9] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: The Penn Treebank," *Comput. Linguist.*, vol. 19, no. 2, pp. 313–330, Jun. 1993.
- [10] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.
- [11] H. Sultanov, "Application of swarm and reinforcement learning techniques to requirements tracing," Ph.D. dissertation, University of Kentucky, 2013.
- [12] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Transactions on Software Engineering*, vol. 28, no. 10, pp. 970–983, Oct. 2002.
- [13] J. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Proc. of the 11th IEEE Int. Requirements Engineering Conf.*, Sep. 2003, pp. 138–147.
- [14] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using Latent Semantic Indexing," in *Proc. of the 25th Int. Conf. on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 125–135.
- [15] H. yi Jiang, T. Nguyen, I.-X. Chen, H. Jaygarl, and C. Chang, "Incremental latent semantic indexing for automatic traceability link evolution management," in *Proc. of the 23rd IEEE/ACM Int. Conf. on Automated Software Engineering*, Sep. 2008, pp. 59–68.
- [16] S. Pandanaboyana, S. Sridharan, J. Yannelli, and J. Hayes, "Requirements tracing on target (retro) enhanced with an automated thesaurus builder: An empirical study," in *Proc. of the Int. Workshop on Traceability in Emerging Forms of Software Engineering*, May 2013, pp. 61–67.
- [17] X. Zou, R. Settimi, and J. Cleland-Huang, "Improving automated requirements trace retrieval: a study of term-based enhancement methods," *Empirical Software Engineering*, vol. 15, no. 2, pp. 119–146, 2010.
- [18] N. Niu and S. Easterbrook, "Extracting and modeling product line functional requirements," in *Proc. of the 16th IEEE Int. Requirements Engineering Conf.*, Sep. 2008, pp. 155–164.
- [19] G. Spanoudakis, A. Zisman, E. Prez-miana, and P. Krause, "Rule-based generation of requirements traceability relations," *Journal of Systems and Software*, vol. 72, pp. 105–127, 2004.
- [20] H. Sultanov, J. H. Hayes, and W.-K. Kong, "Application of swarm techniques to requirements tracing," *Requir. Eng.*, vol. 16, no. 3, pp. 209–226, Sep. 2011.
- [21] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," in *Proc. of the 32nd ACM/IEEE Int. Conf. on Software Engineering*, vol. 1, New York, NY, USA, 2010, pp. 155–164.
- [22] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms," in *Proc. of the 2013 Int. Conf. on Software Engineering*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 522–531. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2486788.2486857>
- [23] S. Winkler, "Trace retrieval for evolving artifacts," in *Proc. of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, ser. TEFSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 49–56. [Online]. Available: <http://dx.doi.org/10.1109/TEFSE.2009.5069583>
- [24] —, *EvoTrace: Evolution-Aware Trace Retrieval*. Hamburg, Germany: Verlag Dr. Kovac, 2011.
- [25] P. Mäder and O. Gotel, "Towards automated traceability maintenance," *Journal of Systems and Software*, vol. 85, no. 10, pp. 2205 – 2227, 2012, automated Software Evolution. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0164121211002779>
- [26] J. Cleland-Huang, P. Mader, M. Mirakhori, and S. Amornborvornwong, "Breaking the big-bang practice of traceability: Pushing timely trace recommendations to project stakeholders," in *Proc. of the 20th IEEE Int. Requirements Engineering Conf. (RE)*, Sept 2012, pp. 231–240.
- [27] S. Lohar, S. Amornborvornwong, A. Zisman, and J. Cleland-Huang, "Improving trace accuracy through data-driven configuration and composition of tracing features," in *Proc. of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, pp. 378–388. [Online]. Available: <http://doi.acm.org/10.1145/2491411.2491432>

How Do Users Like This Feature?

A Fine Grained Sentiment Analysis of App Reviews

Emitza Guzman
Technische Universität München
Garching, Germany
emitza.guzman@mytum.de

Walid Maalej
University of Hamburg
Hamburg, Germany
maalej@informatik.uni-hamburg.de

Abstract—App stores allow users to submit feedback for downloaded apps in form of star ratings and text reviews. Recent studies analyzed this feedback and found that it includes information useful for app developers, such as user requirements, ideas for improvements, user sentiments about specific features, and descriptions of experiences with these features. However, for many apps, the amount of reviews is too large to be processed manually and their quality varies largely. The star ratings are given to the whole app and developers do not have a mean to analyze the feedback for the single features. In this paper we propose an automated approach that helps developers filter, aggregate, and analyze user reviews. We use natural language processing techniques to identify fine-grained app features in the reviews. We then extract the user sentiments about the identified features and give them a general score across all reviews. Finally, we use topic modeling techniques to group fine-grained features into more meaningful high-level features. We evaluated our approach with 7 apps from the Apple App Store and Google Play Store and compared its results with a manually, peer-conducted analysis of the reviews. On average, our approach has a precision of 0.59 and a recall of 0.51. The extracted features were coherent and relevant to requirements evolution tasks. Our approach can help app developers to systematically analyze user opinions about single features and filter irrelevant reviews.

I. INTRODUCTION

Application distribution platforms, or app stores, allow users to search, buy, and deploy software apps for mobile devices with a few clicks. These platforms also allow users to share their opinion about the app in text reviews, where they can, e.g., express their satisfaction with a specific app feature or request a new feature. Recent empirical studies [4], [6], [21] showed that app store reviews include information that is useful to analysts and app designers, such as user requirements, bug reports, feature requests, and documentation of user experiences with specific app features. This feedback can represent a "voice of the users" and be used to drive the development effort and improve forthcoming releases [15], [25].

However, there are several limitations which prevent analysts and development teams from using the information in the reviews. First, app stores include a *large amount* of reviews, which require a large effort to be analyzed. A recent study found that iOS users submit on average 22 reviews per day per app [21]. Very popular apps such as Facebook get more than 4000 reviews per day. Second, the *quality* of the reviews varies widely, from helpful advice and innovative ideas to insulting comments. Third, a review typically contains a *sentiment mix*

concerning the different app features, making it difficult to, e.g., filter positive and negative feedback or retrieve the feedback for specific features. The usefulness of the star ratings in the reviews is limited for development teams, since a rating represents an average for the whole app and can combine both positive and negative evaluations of the single features.

To reduce the effort spent in collecting and understanding user feedback from the app reviews, we propose an approach that automatically extracts app features referred in the reviews together with the user opinions about them. We refer to a feature as a prominent or distinctive visible characteristic or quality of an app [10]. It can be any description of specific app functionality visible to the user (e.g., "uploading files" or "sending a message"), a specific screen of the app (e.g., "configuration screen"), a general quality of the app (e.g., "load time", "size of storage", or "price"), as well as specific technical characteristics (e.g., "encryption technology").

Our approach produces a fine-grained list of features mentioned in the reviews. It then extracts the user sentiments of the identified features and gives them a general score across all reviews. Finally, it groups fine-grained features into more meaningful high-level features that tend to be mentioned in the same reviews and shows the opinions of users about these high-level features. We use collocation finding [17] for extracting the fine-grained features, sentiment analysis [28] for extracting the sentiments and opinions associated to the features, and topic modeling [2] for the grouping of related features.

We evaluated the approach with 32210 reviews for seven iOS and Android apps and compared the results with 2800 manually peer-analyzed reviews. The results show that our approach successfully extracts the most frequently mentioned features, that the groups of features are coherent and relevant to app requirements, and that the sentiment analysis results positively correlate to the manually assigned sentiment scores.

The contribution of this paper is threefold. First, we introduce an approach to automatically extract fine-grained and coarse-grained features from the text of app reviews. Second, we present a method for aggregating the sentiments of many users for a feature by applying automated sentiment analysis techniques on app reviews. Finally, we describe a detailed evaluation method based on content analysis, a manually created evaluation dataset, and an evaluation tool, which can all be used for similar evaluations.

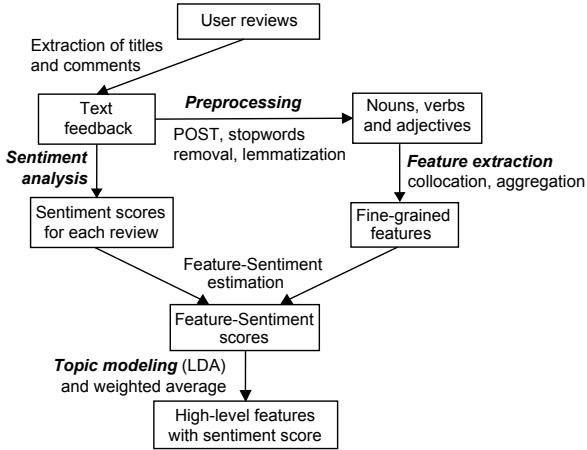


Fig. 1: Overview of the approach.

The remainder of the paper is structured as follows. Section II introduces the approach. Section III describes the evaluation method and the content analysis study. Section IV reports on the results. Section V discusses the findings, their implications, and limitations. Finally, Section VI summarizes the related work, while Section VII concludes the paper.

II. APPROACH

The main goal of our approach is to automatically identify application features mentioned in user reviews, as well as the sentiments and opinions associated to these features. For this we use Natural Language Processing, and Data Mining techniques. Figure 1 shows an overview of the approach. First, we collect the user reviews for a specific app and extract the title and text comments from each review. Then, we *preprocess* the text data to remove the noise for feature extraction. Afterwards, we extract the features from the reviews by applying a *collocation finding* algorithm and aggregating features by their meaning. This produces the list of fine-grained features, each consisting of two keywords. To extract the sentiments about the feature we apply *lexical sentiment analysis* to the raw data from the titles and comments. Lexical sentiment analysis assigns a sentiment score to each sentence in the review. When a feature is present in the sentence, the sentiment score of the sentence is assigned to the feature. Finally, we apply a *topic modeling* algorithm to the extracted features and their associated sentiment scores to create a more coarse-grained summary, which groups features that are mentioned in the same reviews. In the following we explain the main steps of our approach.

A. Data Collection and Preprocessing

In the first step we collect and preprocess the user reviews. When developing and evaluating our approach we used reviews from the Apple App Store¹ and Google Play². However, the approach can also be applied to reviews from other platforms. For collecting the Apple Store data we use a modified version

of an open source scraping tool³. For the collection of Google Play reviews we developed a tool which uses the Google Play Application Programming Interface (API). We store the collected data in a MySQL database. After gathering the data we extract the title and comments from each review as both might include references to features or sentiments.

While the sentiment analysis does not require further preprocessing, the feature extraction requires three additional steps:

- **Noun, verb, and adjective extraction.** We use the part of speech (POS) tagging functionality of the Natural Language Toolkit, NLTK⁴, for identifying and extracting the nouns, verbs, and adjectives in the reviews. We assume that these parts of speech are the most likely to describe features as opposed to others such as adverbs, numbers, or quantifiers. A manual inspection of 100 reviews confirmed this assumption.
- **Stopword removal.** We remove stopwords to eliminate terms that are very common in the English language (e.g., “and”, “this”, and “is”). We use the standard list of stopwords provided by Lucene⁵ and expand it to include words that are common in user reviews, but are not used to describe features. The words we added to the stopword list are the name of the application itself, as well as the terms “app”, “please”, and “fix”.
- **Lemmatization.** We use the Wordnet [19] lemmatizer from NLTK for grouping the different inflected forms of words with the same part of speech tag which are syntactically different but semantically equal. This step reduces the number of feature descriptors that need to be inspected later. With this process, for example, the terms describing the verbs “sees” and “saw” are grouped into the term “see”.

B. Feature Extraction

We use the collocation finding algorithm provided by the NLTK toolkit for extracting features from the user reviews. A collocation is a collection of words that co-occur unusually often [1]. Manning and Schütze define collocations as expressions of two or more words which correspond to a conventional way of referring to things [17]. An example of a collocation is *<strong tea>*, whereas *<powerful tea>* is not a collocation since these two words are not normally used in the English language together. Collocations do not necessarily imply that the words are adjacent. Several words can be between the words that constitute the collocation e.g., *<knock door>*. Features can generally be described as collocations, as they are normally a collection of terms that are used repeatedly to convey a specific meaning. Examples of collocations that identify application features are *<pdf viewer>*, *<user interface>* and *<view picture>*. We use a likelihood-ratio test [17] for finding collocations consisting of two words in our reviews.

¹<https://itunes.apple.com/us/genre/ios/id36>

²<https://play.google.com/store?hl=en>

³<https://github.com/oklahomaok/AppStoreReview>

⁴<http://nltk.org/>

⁵<https://lucene.apache.org/>

TABLE I: Examples of SentiStrength scores in the user reviews.

| Sentence in review | Word scores | Sentence score |
|---|--|----------------|
| had fun using it before but now its really horrible :(help!! | had fun[2] using it before but now its really horrible[-4] [-1 booster word] :([-1 emoticon] help![+1 punctuation emphasis] | {2, -5} |
| uploading pictures with the app is so annoying! | uploading pictures with the app is so annoying[-3]! [-1 punctuation emphasis] | {1, -3} |
| pleeeeease add an unlike button and I will love you forever!! | pleeeeease[3] [+0.6 spelling emphasis] add an unlike button and I will love[3] you forever!![+1 punctuation emphasis] | {5, -1} |

After finding the collocations we filter them by taking into consideration only those that appear in at least three reviews and that have less than three words (nouns, verbs, or adjectives) distance between them. Typically, the order of the words is important for collocations. However, we consider word ordering unimportant for describing features. For example, we consider that the pairs of words *<pdf viewer>* and *<viewer pdf>* convey the same meaning, although the latter might be used less frequently. Therefore, we merge collocations that consist of the same words but have different ordering.

Users can use different words to refer to the same features. We group collocations whose pairs of words are synonyms and use Wordnet as a synonym dictionary. Wordnet also allows us to group collocations with misspellings together.

When grouping features together we consider the word collection with the highest frequency to be the name of the feature. For example, if we have the following word collections: *<picture view>*, *<view photographs>* and *<see photo>* with a frequency of 30, 10, and 4 respectively. Our approach would then group these features together since they are synonyms. Afterwards, it would choose the one with the highest frequency as the name for the feature, in this case *<picture view>*.

C. Sentiment Analysis

Sentiment analysis is the process of assigning a quantitative value (positive or negative) to a piece of text expressing an affect or mood [12]. For analyzing sentiments in user reviews, we use SentiStrength [28], a lexical sentiment extraction tool specialized in dealing with short, low quality text. Previous research has shown that SentiStrength has a good accuracy for short texts in social media, such as, Twitter and movie reviews [27]. Pagano and Maalej [21] found that 80.4% of the comment reviews in the App Store contain less than 160 characters, making SentiStrength a good candidate for analyzing sentiments in user reviews.

SentiStrength divides the review text into sentences and then assigns a positive and negative value to each sentence. The idea behind this is that humans can express both positive and negative sentiments in the same sentence, e.g., "I loved the app until this last release, it's awful now". SentiStrength assigns positive scores in the $[+1, +5]$ range, where +5 denotes an extremely positive sentiment and 1 denotes the absence of sentiment. Similarly, negative sentiments range from $[-1, -5]$, where -5 denotes an extremely negative sentiment and -1 indicates the absence of any negative sentiment.

Furthermore, SentiStrength assigns fixed scores to tokens in a dictionary where common emoticons are also included. For example, "love" is assigned a score of $\{3, -1\}$ and "hate" a $\{1, -4\}$ score. Only words that are present in the dictionary are attributed with an individual score. Modifier words and symbols also alter the score. For example, "absolutely love" is assigned a score of $\{4, -1\}$. The same score is given to "looove" and "love!!!". Additionally, it offers the possibility to extend its dictionary. However, for our approach we used the dictionary provided by the tool. The sentiment score of the whole sentence is computed by taking the maximum and minimum scores among all the words in a sentence. Table I shows three examples of sentences scored by SentiStrength.

After calculating the sentiment score in the sentences, we compute the sentiment score for the features. We consider the sentiment score of a feature to be equal to the positive or negative score of the sentence in which it is present. As a feature score we choose the score with the maximum absolute value. In the case that the positive and negative values are the same, we assign the negative value to the feature. For example, lets consider the sentence "Uploading pictures with the app is so annoying!". This sentence contains the feature *<uploading pictures>*, and the sentiment score of the sentence is $\{1, -3\}$. Therefore, we assign the feature the sentiment score of -3. This step produces a list of all extracted features, their frequencies (how often they were mentioned), and their sentiment scores. This list is a fine-grained summary which developers and requirement analysts can use to get a detailed overview of users' primary concerns.

D. Topic Modeling

Our approach produces a high-level summary as a final result. This summary contains groups of different features and a corresponding sentiment score. To group features that tend to co-occur in the same reviews we use Latent Dirichlet Allocation (LDA) [2], a topic modeling algorithm. LDA is a probabilistic distribution algorithm which uses Gibbs sampling to assign topics to documents, in our case user reviews. In LDA a topic is a probabilistic distribution over words and each document is modeled as a mixture of topics. This means that each review can be associated to different topics and that topics are associated to different words with a certain probability. An example of a topic can be the set of words *{crash, update, frustrated, newest, version, help, bug}* which describes users' experiences when updating a faulty app.

We used the Matlab Topic Modeling Toolbox⁶. Instead of inputting the words forming the vocabulary of our analyzed reviews to the LDA algorithm, as is normally the case when applying LDA, we input the list of extracted features and model each feature as a single word. For example, the feature described with the *<picture view>* collocation is transformed into the single term *picture_view*. LDA then outputs the feature distribution of each topic and the probabilistic topic composition for each review. An example of a topic with this modification could then be the set of features *{picture_view, camera_picture, upload_picture, delete_picture}* which describes features related to manipulating pictures in an application.

We calculate topic sentiments as follows. Let $R = \{r_1, r_2, \dots, r_n\}$ be the set of analyzed reviews and $T = \{t_1, t_2, \dots, t_m\}$ the set of extracted topics. The final output of the LDA computation is the matrix $W_{n \times m}$, where $w_{i,j}$ contains the number of times a feature mentioned in review r_i is associated with topic t_j . We then use a weighted average to calculate the sentiment score of each topic. For every topic t_j we calculate the topic sentiment score ts_j as:

$$ts_j = \frac{\sum_{i=1}^n w_{i,j} \cdot s_i}{\sum_{i=1}^n w_{i,j}}$$

where $S = \{s_1, s_2, \dots, s_l\}$ denotes the sentiment score of each feature associated to the topic t_j .

III. EVALUATION METHOD

Our evaluation goal was twofold. We aimed at evaluating (a) the relevance of the automatically identified features from the reviews and (b) the correctness of the automatically calculated sentiment estimation for each feature over all reviews. The specific evaluation questions were:

- 1) Does the extracted text represent real app features?
- 2) Are the extracted and grouped features coherent and relevant for app analysts and developers?
- 3) Is the automated sentiment estimation similar to a manually conducted sentiment assessment?

To answer these questions, we created a truth set through a careful, manual, peer-conducted content analysis process [20]. We then compared the results of our approach against the manual analysis. In the following we describe the dataset that was used in our evaluation, how the truth set was created, as well as the quality metrics that we considered for the evaluation. To encourage replication, we make the evaluation data and evaluation tools publicly available on the project website⁷.

A. Dataset

Our evaluation data consisted of user reviews from the US App Store and Google Play. The App Store is an application

TABLE II: Overview of the evaluation apps.

| App | Category | Platform | #Reviews | ØLength |
|-------------|---------------|-------------|----------|---------|
| AngryBirds | Games | App Store | 1538 | 132 |
| Dropbox | Productivity | AppStore | 2009 | 172 |
| Evernote | Productivity | App Store | 8878 | 200 |
| TripAdvisor | Travel | App Store | 3165 | 142 |
| PicsArt | Photography | Google Play | 4438 | 51 |
| Pinterest | Social | Google Play | 4486 | 81 |
| Whatsapp | Communication | Google Play | 7696 | 38 |

distribution platform for Apple devices, whereas Google Play distributes applications for Android devices. Both app stores allow users to write reviews about the downloaded apps. Additionally, both stores cluster the apps into different categories based on functionality. For our evaluation, we selected seven apps from the lists of “most popular apps” in different categories. Table II shows these apps, their categories, and the number of reviews considered in the evaluation.

For the App Store we selected the apps AngryBirds, Dropbox, Evernote, and TripAdvisor from the categories Games, Productivity, and Travel. We collected all user reviews written in 2013 for these apps. For Google Play we selected the apps PicsArt, Pinterest, and Whatsapp from the categories Photography, Social, and Communication. We collected the maximum number of reviews allowed by the Google Play APIs. These were the most recent 4000 to 7000 reviews for the apps, as of February 2014.

We selected different categories of apps because we wanted to evaluate our approach against reviews containing diverse vocabularies, describing different features, and written by different user audiences. Users from Angrybirds, Dropbox, and TripAdvisor might have different expectations, interact with technology in various manners, belong to different age groups, and express their sentiments and experiences in different ways.

We chose popular apps to increase the probability that the people creating the truth set were familiar with the apps, reducing the manual feature extraction effort and minimizing errors during the truth set creation. Popular apps are also more likely to have more reviews. An automated analysis for these apps would probably be more realistic and useful. On average, the reviews of the App Store apps had 178 characters, while Google Play apps included only 53 characters.

For each user review we collected the title, comment, date, author, version, and star rating. We ran our approach on the text in the title and comment of each review. To compare the results generated by our approach with human assessments, we created a truth set of the features mentioned in the review and their associated sentiments.

B. Truth Set Creation

The methodological cornerstone for the creation of the truth set was the use of content analysis techniques as described by Neuendorf [20] and by Maalej and Robillard [16]. This process involved the systematic assessment of a reviews sample by

⁶http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm

⁷<http://mobilis.informatik.uni-hamburg.de/app-review-analysis/>

human coders, who read each review and assessed its contents according to a strict coding guide. Apart from both authors, this process involved seven trained coders who independently coded 2800 randomly sampled user reviews totaling 60,738 words. For each user review, two coders independently:

- 1) indicated whether the review contained a feature request or feedback about an existing feature,
- 2) identified the app features mentioned in the review, and
- 3) assessed the sentiments associated to each feature.

The first step of the truth set creation consisted of developing the coding guide. We needed the guide because the review content and the coding task can be interpreted differently by distinct coders. The goal of a coding guide is to systemize the task and minimize disagreements between peer-coders. The guide contained instructions about the task, clear definitions of a feature, feature request, feedback on a feature, and of the different sentiment scales. The guide also included examples for each possible assessment and rules to follow. The coding guide was created in an iterative process including four main iterations. In each iteration, we tested the guide by independently peer-assessing 50 reviews.⁸ The coders' disagreement was manually analyzed and the coding guide was modified (e.g., include more examples and improve the definitions) to avoid similar disagreements in the next iterations.

The second step consisted of the sampling. We selected 400 reviews for each of the evaluation apps based on stratified random sampling [23]. Our sampling scheme took into account the rating distribution of the specific apps. For instance, assume that the app has 1000 reviews, the sample size is of 100, and the rating distribution in all reviews is, e.g., as follows: 40% of the reviews had 5 stars, 40% had 4 stars, 15% had 3 stars, 5% had 2 stars and 1% had one star. Then, the sample with 100 reviews includes 40 reviews of 5 stars, 40 of 4 stars, 15 of 3 stars, etc. The reviews in each strata were selected randomly from the corresponding group of ratings.

The third step consisted of peer-coding the reviews in the samples. For this task, we developed a coding tool, which displays a single review (title, comment, and rating) at a time together with the coding questions. Figure 2 shows a screenshot of the tool. When using the coding tool the coders could select the features from the review text by clicking on the words describing the feature. Additionally, the coders assigned a sentiment to each of the features. The sentiment scale was very positive, positive, neutral, negative, and very negative. Furthermore, the coders were asked to indicate if the review included feedback about an already existing feature, a request for a new feature, a bug report, or other type of content. Multiple selections were allowed. Coders were able to stop and resume their coding tasks at any time they wished.

The coders were graduate students from the Technical University of Munich and the University of Hamburg. They all had a high command of English and had software development experience. The reviews were randomly assigned to the coders,

so that each review was assigned twice and that each coder shared a similar number of assignments with all other coders.

Each coder received the coding guide, the tool, and the coding assignments. Moreover, we explained the coding task in a short meeting and we were available for clarification requests during the coding period. To assure that the coders had some knowledge about the evaluation apps, we asked them to read the app store descriptions of each of the seven apps.

We also asked the coders to record the total time spent on their coding assignments in order to estimate the amount of effort necessary for a fine-grained manual analysis of user reviews. Coders reported spending between 8 and 12.5 hours for coding about 900 reviews. These numbers confirm previous studies, which have highlighted the large amount of effort needed to manually analyze user feedback [4], [21].

The final step in the truth set creation consisted of the analysis of disagreements. Overall 30% of the coded reviews included feedback about features, while 10% included feature requests. The average sentiment for all coded features was 0.17 (i.e., neutral). These results confirm the findings of previous exploratory studies [21]. The coders identified a total of 3005 features, agreeing on 1395 features (53%) and disagreeing on 1610 features (47%). The coders also agreed that 1086 reviews did not contain any features. This disagreement was handled in two steps. First, we randomly selected 100 reviews with disagreements and analyzed reasons, and common patterns for falsely classified features. We then used the results to automatically filter misclassified features. Second, one of the authors manually reviewed the remaining features with disagreement and decided if each of the previously labeled features was mentioned in the review or not. At the end, the truth set included 2928 features. We solved the disagreement between the sentiments associated to features by transforming the categorical values into numerical values and calculating the average of both coders.

C. Evaluation Metrics

We evaluate our topics using precision, recall, and F-measure. Precision is computed by dividing the number of true positives by the sum of true positives and false positives. Recall is computed by dividing the number of true positives by the sum of true positives and false negatives. We use the general form of the F-measure, which combines the precision and recall results, for its computation.

We define a feature as true positive, if it was automatically extracted from a review and was also manually identified in that review. False positives are features that were automatically associated to a review in one of the topics, but were not identified manually in that review. Finally, false negative features were manually identified in a review but were not present in any of the extracted topics associated to the review.

In addition to precision, recall, and F-measure we used two further metrics for determining the quality of the topics (groups of features) generated by the LDA algorithm. The *coherence* of topics assesses how well the topics are logical and consistent and whether they share a common theme. The *requirements*

⁸The test reviews were different from the reviews in the evaluation samples.

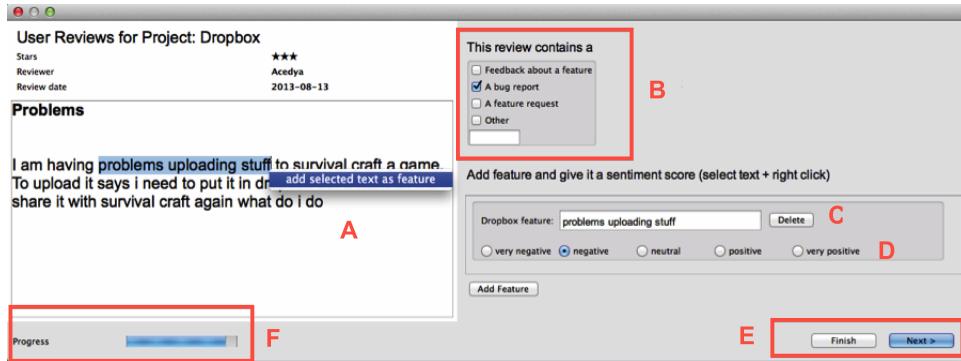


Fig. 2: Coding tool for the creation of the truth set (available on the project website). A: the review text to be coded, B: the type of the review, C: features references in the review, D: sentiment about the feature, E: navigation button, and F: progress bar.

relevance measures whether the topics contain information that help define, understand, and evolve the requirements of the app. These last two metrics were qualitatively evaluated by the authors with a 5-level scale.

IV. EVALUATION RESULTS

We first report on the feature extraction results and then on the evaluation of the sentiment estimations.

A. Feature Extraction

We evaluate the high-level topics generated by our approach and not the fine-grained list of features. The topics contain the fine-grained features, therefore, the results from the topic evaluation reflect the performance of the fine-grained feature extraction. Due to the large amount of fine-grained features (e.g., 3700 for Evernote) a manual qualitative evaluation would be unfeasible and the consideration of the N-top features would produce unwanted bias.

After running the feature extraction step of our approach we obtained a list of wordsets designating app features. Many of the wordsets extracted as features contained words that do not describe features but rather the opinions or sentiments of users (e.g., great, bad, good, like, hate...). To filter these words we decided to slightly modify our approach and include all words that are assigned a sentiment by the lexical sentiment analysis tool into the stopword list. We use F_S to refer to the original approach which includes words with a sentiment meaning into the feature extraction algorithm, and F_{NS} to refer to the modified approach which excludes sentiment words when generating feature descriptors. Examples of the most common features extracted by the F_{NS} approach for two of the applications are shown in Figure 3.

Table III shows the number of different features extracted for each app for the original approach F_S and for the modified approach F_{NS} . The number of extracted feature varies between 181 and 3700 features. To deal with this large amount of information, we gather the extracted features into high-level groups using topic modeling.

We calculated the precision, recall and F-measure of the extracted topic models for the seven evaluation apps. We generated 20 topic models for each app. Table V and Table VI

TABLE III: Number of extracted fine-grained features per app.

| App | F_S | F_{NS} |
|-------------|-------|----------|
| AngryBirds | 284 | 219 |
| Dropbox | 612 | 600 |
| Evernote | 3700 | 3127 |
| TripAdvisor | 846 | 754 |
| PicsArt | 290 | 181 |
| Pinterest | 625 | 465 |
| Whatsapp | 383 | 234 |

TABLE IV: Precision, Recall, and F-measure for the high-level feature extraction with topic modeling.

| App | Precision | Recall | F-measure |
|------------------------------------|--------------|--------------|--------------|
| | F_S | | |
| AngryBirds | 0.335 | 0.332 | 0.334 |
| Dropbox | 0.608 | 0.475 | 0.533 |
| Evernote | 0.474 | 0.416 | 0.443 |
| TripAdvisor | 0.421 | 0.399 | 0.410 |
| PicsArt | 0.750 | 0.669 | 0.707 |
| Pinterest | 0.644 | 0.623 | 0.634 |
| Whatsapp | 0.843 | 0.728 | 0.781 |
| F_S Average | 0.582 | 0.520 | 0.549 |
| | | | F_{NS} |
| | | | |
| AngryBirds | 0.368 | 0.321 | 0.343 |
| Dropbox | 0.603 | 0.473 | 0.531 |
| Evernote | 0.451 | 0.389 | 0.418 |
| TripAdvisor | 0.403 | 0.370 | 0.386 |
| PicsArt | 0.815 | 0.661 | 0.730 |
| Pinterest | 0.658 | 0.592 | 0.623 |
| Whatsapp | 0.910 | 0.734 | 0.813 |
| F_{NS} Average | 0.601 | 0.506 | 0.549 |

show examples of topics and their associated sentiments for the Dropbox and Pinterest apps respectively. We compared the results when using the F_{NS} approach which excludes words with sentiments from the feature extraction, and F_S which includes them. Table IV summarizes the results. Both approaches had similar results, varying on a project basis. We achieved the highest precision of 91% for Whatsapp with F_{NS} and the highest recall of 73% for the same app with F_{NS} . On

TABLE V: Most common topics extracted from the user reviews of the Dropbox app with their sentiments

| Topic | Senti. score |
|--|----------------------|
| upload_photo, load_photo, photo_take, photo_want, move_photo, keep_upload, keep_try | 1.51 <i>Positive</i> |
| file_name, folder_file, rename_file, file_add, folder_rename, make_folder, change_name, file_copy, option_folder | 1.49 <i>Positive</i> |
| file_load, video_load, video_upload, download_file, download_video, download_phone, download_time, file_phone, update_need, computer_phone | 1.75 <i>Positive</i> |

TABLE VI: Most common topics extracted from the user reviews of Pinterest with their sentiments

| Topic | Senti. score |
|--|---------------------------|
| board_pin, pin_wish, make_board, create_board, sub_board, create_pin, use_board, edit_board, button_pin, edit_pin | 2.47 <i>Very Positive</i> |
| pin_see, pin_go, load_pin, keep_thing, board_change, pin_scroll, click_pin, pin_everything, time_search, browse_pin | 2.28 <i>Very Positive</i> |
| craft_idea, craft_recipe, home_idea, idea_diy, look_idea, everything_want, home_decor, thing_internet, art_craft, load_image | 2.23 <i>Very Positive</i> |

average, the precision of F_{NS} was approximately 60% and the recall was about 50%. The average precision of F_S was 58% and its recall was of 52%. We observed the lowest precision and recall for AngryBirds, an iOS game, which also resulted in the largest amount of disagreement during the truth set creation. Android Apps had a higher precision and recall than iOS apps.

For measuring the coherence and the requirements relevance, we manually examined the 20 topics generated by the F_{NS} version of our approach for each evaluation app. We also examined the 10 features which were most strongly associated to each of the topics. Table VII summarizes the results.

Coherence: To qualitatively measure the coherence of the topics generated by our approach we manually analyzed the ten most popular features for each topic. We evaluated the coherence of each topic by analyzing if the features conforming the topic shared a common theme. Then, we rated the coherence of each topic on a 5-level scale (from *very good* to *very bad*). Afterwards, we converted the individual ratings for each topic into a numerical scale ($[-2,2]$ range) and calculated a coherence average for each app. The approach had a *good* to *neutral* coherence for all apps with the exception of Whatsapp. The difference in the coherence levels for Whatsapp can be explained by the average length of its reviews. These were much shorter than for the reviews of the other apps and LDA has a difficulty for generating meaningful topics on sparse data [26]. With the exception of Whatsapp, mixed topics (topics with no common theme) were not prevalent in the other apps.

TABLE VII: Coherence and requirements engineering relevance of topics.

| App | Coherence | Req. Relevance |
|-------------|-----------|----------------|
| AngryBirds | Good | Good |
| Dropbox | Good | Very Good |
| Evernote | Good | Good |
| TripAdvisor | Good | Very Good |
| PicsArt | Neutral | Good |
| Pinterest | Good | Good |
| Whatsapp | Bad | Good |

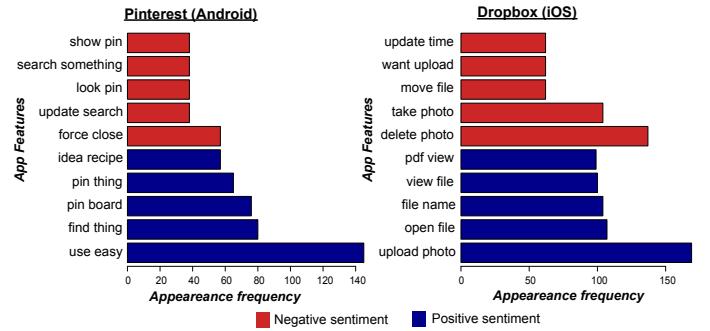


Fig. 3: Extracted features from Pinterest and Dropbox apps. Positive features are represented in blue, negative in red.

However, the presence of duplicate topics (topics sharing a very similar theme) was more prevalent in apps with less functionality or shorter length reviews, such as PicsArt and Whatsapp. LDA allows for the configuration of the number of topics and more experimentation with this variable could lead to less duplicate topics and different coherence results.

Requirements relevance: We evaluated the relevance of the extracted topics to the app requirements by manually analyzing the 10 most popular extracted features for each topic. Similarly to the coherence evaluation, we used a 5-level scale to rate whether a topic was relevant for requirements engineering. We considered a topic to be relevant to requirements when it mostly consisted of app features, app qualities, or information indicating how the users utilized the app.

Our approach generated topics with a *very good* to *good* relevance to requirements engineering for all apps. Even when the topics were not coherent, the 10 most popular extracted features were usually words describing actual app features and were important for understanding how users use the app. Each app usually had one or two topics with words that describe bug reports. These wordsets were usually not considered as features in our manual coding. However, they might contain valuable information for the evolution and maintenance of the app. The presence of noise (false positives) in the 10 most popular features (topics) was virtually non-existent for all apps.

B. Sentiment Analysis

The approach proposed in this work assigns a feature the sentiment score of the sentence in which it is located. Figure 3

shows examples of extracted features and their associated sentiment. This information can be used to detect the features with the highest and lowest user approval. To compare the feature sentiment scores extracted by our approach with the ones given by the coders we converted the automatically extracted scores to categorical values. Similar to Kucuktunc et al. [12] we consider all reviews in the (2,5] range to be *very positive*, those in the (1,2] range *positive*, whereas those in the [-1,1] range are *neutral*. Reviews with a sentiment score in the [-2,-1) range are considered *negative* and those with a [-5,-2) range *very negative*. In this way we converted all sentiments computed by our approach into the scale used by the coders.

We then converted the categorical values into numerical values in the [-2,2] range, where -2 denotes a very negative sentiment and 2 a very positive sentiment. After that, we calculated the average sentiment score given by the two coders and used this as the “true sentiment” associated to each feature.

The Spearman’s rho correlation coefficient between the sentence-based sentiment score and the truth set was of 0.445 (p-value< 2.2e-16), indicating a moderate positive correlation between them. Figure 3 shows examples of the most common positive and negative features for Dropbox and Pinterest.

We calculated an additional sentiment score for each feature by assigning each feature the sentiment score of the whole review in which it is mentioned. We compute the sentiment of an entire review by calculating the positive and negative average scores of all sentences in the review separately. For the case where both positive and negative sentence averages are in the [-1,1] range we assign the whole review the neutral score of 0. When the sentence negative average multiplied by 1.5 is less than the positive average, we assign the review the sentiment score of the negative average. In the opposite case, the review is assigned the positive average score. When comparing the positive and negative sentence average we multiply by 1.5 because negativity is considered to be less frequent in human written texts.⁹

With this review-based sentiment estimation, the positive correlation between the sentiment score and the truth set sentiment score was of 0.592 (p-value< 2.2e-16), representing a strong positive correlation.

One possible explanation for the higher correlation of the review-based sentiment score in comparison with the sentence-based approach is that people frequently use more than one sentence to express their opinion about a specific feature. That is, context is important when determining the sentiment associated to a specific feature.

V. DISCUSSION

In this section we discuss our results and describe the limitations, threats to validity, and implications of our approach.

A. Result Interpretation

The qualitative and quantitative results of our approach are promising. Our approach was able to detect app features

mentioned in the user reviews for a diverse set of apps belonging to different categories and serving different types of users. The feature extraction approach had a *good* performance for apps belonging to all analyzed categories with exception of the games category. One possible explanation can be the various ways users describe the gaming app features. Human coders had the largest difficulties when identifying features from this type of apps, as noted by their level of disagreement, confirming the challenges of examining this type of reviews.

The qualitative evaluation showed that the topics were coherent, contained little noise, and were relevant for requirements engineering tasks. Even for the apps where the recall was relatively low (i.e., 33-47%) the qualitative results showed that the topics accurately describe the overall functionality of the apps. Due to the inner workings of the LDA algorithm [2] the approach generates more coherent topics when dealing with lengthier reviews. Additionally, duplicate topics are more common for apps with less functionality and shorter reviews, such as PicsArts or Whatsapp. An advantage of our approach is that the number of topics (a parameter for LDA) can be manually tuned for each app by the project team to get less duplicate topics and better coherence and precision.

Finally, precision results were higher for apps with short reviews containing few or no features. One important finding is that our approach has a high performance for detecting reviews with no features mentioned in short reviews. This makes the approach useful for filtering non-informative reviews, which, e.g., only include praise or dispraise. These types of reviews tend to be very frequent in app stores [21]. Filtering them will help developers focus on the relevant and informative reviews.

B. Limitations

A limitation of our approach is that non frequently mentioned features are often not detected, as reflected by the recall values. This can be improved by including linguistic patterns, which describe the language structure in which features are described. This would allow for the identification of non common features through such patterns in addition to the term frequency.

The used lexical sentiment analysis approach has the disadvantage of a limited handling of negation and conditionals, and no handling of past tense and sarcasm. However, the positive correlations found in both sentiment score approaches suggest that the noise produced by our approach due to this limitation is minimal. The expansion of the dictionary to include jargon common in user reviews, such as “bug”, “crash”, or “please fix!” could enhance the sentiment analysis performance. The outperformance of the review-based sentiment approach over the sentence-based approach confirms the importance of taking context into account when assigning sentiment scores to features.

C. Threats to Validity

The qualitative evaluation of the topic relevance to requirements engineering was done by the authors of the paper and not by actual developers of the apps. This is a threat to validity as the evaluators could be biased or could have incomplete

⁹See SentiStrength user manual <http://sentistrength.wlv.ac.uk/>

knowledge or misunderstandings about the specific information that developers working on these apps need with respect to requirements engineering. Another threat to validity is the high level of disagreement between (47%) coders of what constitutes an app feature. We tried to alleviate disagreement by providing a coding guide with a precise definitions and examples. Furthermore, we identified patterns for common human errors during the coding task. These human errors were a common cause for disagreement.

D. Implications

Our approach can be used to obtain two-level grained summaries of features mentioned in user reviews where the associated sentiment score gives an overall idea about the user's acceptance and evaluation of the features. A usage scenario could be a tool that visualizes the feedback in three levels of granularities: (1) the topics with their associated sentiments, (2) the list of fine-grained features with their corresponding sentiments ordered by appearance frequency, and (3) the reviews mentioning the features. Developers and requirements analysts could navigate between the different granularities obtaining an overview of the features. This could help them decide on necessary changes to the requirements and software. Finally, since a timestamp is associated to each review, the tool could depict the trends of how features and their sentiments evolve over time and how these trends are related to specific app releases.

VI. RELATED WORK

We focus the related work discussion in three areas: user feedback and crowdsourcing requirements, mining user feedback and app store data, as well as feature extraction in software engineering and in other domains.

A. User Feedback and Crowdsourcing Requirements

Seyff [25] and Schneider et al. [24] propose to continuously elicit user requirements with feedback from mobile devices, including information on the application context. We also previously discussed how user feedback could be considered in software development in general [14] [15]. We presented a classification of user feedback types and distinguished between "implicit feedback" and "explicit feedback". In this paper we propose an approach to systematically analyze explicit user feedback, submitted in form of informal text. Our approach helps to identify useful feedback for app analysts and developers, quantitatively evaluating the opinions about the single features, and grouping popular feature requests. We think that our approach will help the crowdsourcing of requirements.

B. Mining User Feedback

User feedback mining has recently attracted the attention of software engineering researchers resulting in several studies, most of them of exploratory nature. Harman et al. [6] analyzed technical and business aspects of apps by extracting app features from the official app descriptions using a collocation and a greedy algorithm for the extraction and grouping of features.

While their feature extraction mechanism is similar to ours, the motivations are different. We are interested in extracting app features and the users sentiments associated to these features to help software teams understand the needs of their users. We therefore mine the features from the reviews and not from the semi-structured app descriptions.

Iacob and Harrison [8] extracted feature requests from app store reviews by means of linguistic rules and used LDA to group the feature requests. Our work is complementary, we are interested in extracting all app features mentioned in reviews, in presenting them in different granularity levels and extracting their associated sentiments.

Galvis Carreño and Winbladh [4] analyzed the use of LDA to summarize user review comments. Their applied model includes Sentiment Analysis, but it is not the focus of their work. Our work is complementary to theirs as we focus on describing user acceptance of features by generating different granularity levels for the extracted features and by proposing mechanisms for aggregating the sentiment on these different levels. This allows for a more detailed and focused view of user feature acceptance.

Li et al. [13] analyzed user reviews to measure user satisfaction. The authors extracted quality indicators from the reviews by matching words or phrases in the user comments with a predefined dictionary, while we use a probabilistic method (likelihood-ratio) for extracting the features. Zou et al. [29] assessed the quality of API's by analyzing user comments on the web. Unlike our approach they focused on extracting a single feature at a time instead of all features.

C. Automated Feature Extraction and Sentiment Analysis

The automatic extraction of features in text documents has been the focus of several requirements engineering researchers. Knauss et al. [11] used a Naïve Bayes approach to extract clarifications in requirements from software team communication artifacts to detect requirements that are not progressing in a project. Their approach makes use of previously tagged data, while we utilize an unsupervised approach. Dumitru et al. [3] and Hariri et al. [5] extracted features from product descriptions to recommend feature implementation for software product lines through text mining algorithms, they then group them together through diffusive clustering. The features are mined by recognizing keywords present in bullet points lists of product descriptions, while we have no structural information indicating the presence of a feature.

While feature extraction and sentiment analysis (also called opinion mining) are relatively new in software and requirements engineering, they have been used in other domains to analyze movie reviews, for analyzing opinions of different products, such as, movies, cameras and desktop software [7], [22] and blogs [18]. Extracting features and sentiments from app stores poses different challenges than when extracting them from other product reviews, as the text in app store reviews tends to be 3 to 4 times shorter [9], having a length that is comparable to that of a Twitter message [21], but posing an additional challenge

in comparison to feature extraction in Twitter messages due to the absence of hashtags.

VII. CONCLUSIONS AND FUTURE WORK

This work presents an approach for extracting app features mentioned in user reviews and their associated sentiments. The approach produces two summaries with different granularity levels. These summaries can help app analysts and developers to analyze and quantify users' opinions about the single app features and to use this information e.g., for identifying new requirements or planning future releases. We generate the summaries by combining a collocation finding algorithm, lexical sentiment analysis, and topic modeling. We obtained a precision up to 91% (59% average) and a recall up to 73% (51% average). The results show that the generated summaries are coherent and contain the most mentioned features in the reviews. We plan to enhance our approach by including linguistic rules for the detection of infrequent features. Furthermore, we plan to evaluate the approach with developers in order to measure its usefulness in industry settings and detect shortcomings. We also plan to enhance the approach with an interactive visualization. Additionally, we plan to address the limitation of the lexical sentiment analysis for detecting sarcasm and context by including the review rating in the computation of the sentiment score.

ACKNOWLEDGMENTS

We thank Christoph Stanik for his support with the Google Play API, and Ghadeer Eresha, Safey Halim, Mathias Ellmann, Marlo Häring, Hoda Naguib, and Wolf Posdorfer for their support in the study. This work was partially supported by the Mexican Council of Science and Technology (Conacyt).

REFERENCES

- [1] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'reilly, 2009.
- [2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022, Mar. 2003.
- [3] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, page 181, New York, New York, USA, May 2011. ACM Press.
- [4] L. V. Galvis Carreño and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. In *ICSE '13 Proceedings of the 2013 International Conference on Software Engineering*, pages 582–591. IEEE Press, May 2013.
- [5] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher. Supporting Domain Analysis through Mining and Recommending Features from Online Product Listings. *IEEE Trans. Software Eng.*, 39(12):1736–1752, 2013.
- [6] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: MSR for app stores. In *Proc. of Working Conference on Mining Software Repositories - MSR '12*, pages 108–111, June 2012.
- [7] M. Hu and B. Liu. Mining opinion features in customer reviews. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining - KDD '04*, pages 755–760. AAAI Press, July 2004.
- [8] C. Iacob and R. Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *MSR '13 Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 41–44. IEEE Press, May 2013.
- [9] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych. Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceeding of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion - TSA '09*, pages 57–64, New York, New York, USA, Nov. 2009. ACM Press.
- [10] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report, DTIC Document, 1990.
- [11] E. Knauss, D. Damian, G. Poo-Caamano, and J. Cleland-Huang. Detecting and classifying patterns of requirements clarifications. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 251–260, Sept. 2012.
- [12] O. Kucuktunc, B. B. Cambazoglu, I. Weber, and H. Ferhatosmanoglu. A large-scale sentiment analysis for Yahoo! Answers. In *Proc. of the International conference on Web search and data mining - WSDM '12*, pages 633–642, Feb. 2012.
- [13] H. Li, L. Zhang, L. Zhang, and J. Shen. A user satisfaction analysis approach for software evolution. In *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*, volume 2, pages 1093–1097. IEEE, 2010.
- [14] W. Maalej, H.-J. Happel, and A. Rashid. When users become collaborators. In *Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications - OOPSLA '09*, page 981, New York, New York, USA, Oct. 2009. ACM Press.
- [15] W. Maalej and D. Pagano. On the Socialness of Software. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 864–871. IEEE, Dec. 2011.
- [16] W. Maalej and M. P. Robillard. Patterns of Knowledge in API Reference Documentation. *IEEE Transactions on Software Engineering*, 39(9):1264–1282, 2013.
- [17] H. Manning, Christopher D., Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.
- [18] Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai. Topic sentiment mixture: modeling facets and opinions in Weblogs. In *Proc. of the 16th international conference on World Wide Web - WWW '07*, pages 171–180, May 2007.
- [19] G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
- [20] K. A. Neuendorf. *The Content Analysis Guidebook*. Sage Publications, 2002.
- [21] D. Pagano and W. Maalej. User feedback in the appstore : an empirical study. In *Proc. of the International Conference on Requirements Engineering - RE '13*, pages 125–134, 2013.
- [22] A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, pages 339–346. Association for Computational Linguistics, Oct. 2005.
- [23] R. L. Rosnow. *Beginning behavioral research: a conceptual primer*. Pearson/Prentice Hall, Upper Saddle River, {N.J}, 6th ed edition, 2008.
- [24] K. Schneider, S. Meyer, M. Peters, F. Schliephacke, J. Mörschbach, and L. Aguirre. *Product-Focused Software Process Improvement*, volume 6156 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, June 2010.
- [25] N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 37–46. IEEE, 2010.
- [26] J. Shao, Y. Wang, X. Deng, S. Wang, et al. Sparse linear discriminant analysis by thresholding for high dimensional data. *The Annals of statistics*, 39(2):1241–1265, 2011.
- [27] M. Thelwall, K. Buckley, and G. Paltoglou. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1):163–173, Jan. 2012.
- [28] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558, Dec. 2010.
- [29] Y. Zou, C. Liu, Y. Jin, and B. Xie. Assessing Software Quality through Web Comment Search and Analysis. In *Safe and Secure Software Reuse*, pages 208–223. Springer, 2013.

Scaling Requirements Extraction to the Crowd

Experiments with Privacy Policies

Travis D. Breaux and Florian Schaub

Institute for Software Research
Carnegie Mellon University
Pittsburgh, Pennsylvania, United States
{breaux, fschaub}@cs.cmu.edu

Abstract—Natural language text sources have increasingly been used to develop new methods and tools for extracting and analyzing requirements. To validate these new approaches, researchers rely on a small number of trained experts to perform a labor-intensive manual analysis of the text. The time and resources needed to conduct manual extraction, however, has limited the size of case studies and thus the generalizability of results. To begin to address this issue, we conducted three experiments to evaluate crowdsourcing a manual requirements extraction task to a larger number of untrained workers. In these experiments, we carefully balance worker payment and overall cost, as well as worker training and data quality to study the feasibility of distributing requirements extraction to the crowd. The task consists of extracting descriptions of data collection, sharing and usage requirements from privacy policies. We present results from two pilot studies and a third experiment to justify applying a task decomposition approach to requirements extraction. Our contributions include the task decomposition workflow and three metrics for measuring worker performance. The final evaluation shows a 60% reduction in the cost of manual extraction with a 16% increase in extraction coverage.

Index Terms—*requirements extraction; natural language; crowdsourcing*

I. INTRODUCTION

Requirements engineering research has long dealt with the challenge of processing natural language texts to facilitate communication, specify systems and conduct requirements validation [34]. Recently, new sources for requirements extraction and analysis have come to include project-specific mailing lists [22], privacy policies [7, 29] and regulatory codes and laws [11, 16], to name a few. The challenge of acquiring requirements from such sources, however, is that each medium carries a native writing style that proficient readers are accustomed to: for example, the style of exchanging e-mails is similar to writing professional letters in which actors interact through text, defining important terms, critiquing positions and adding clarification [22]; whereas, the style of law is more structured with preambles, definitions, sections based on topic and cross-references [5].

Requirements researchers have extracted requirements from text using a combination of manual and automated approaches. Manual approaches may include simple, repeatable steps accompanied by a coding frame that are used to classify the text [35]. To assess validity, researchers apply various forms of inter-rater reliability to coded data sets extracted by multiple, trained analysts. The value of manual

extraction is that, when the method is derived from the dataset, called a *grounded theory* [12], the derivation process surfaces complex nuances and boundary cases that are more likely addressable using human-interpretable heuristics. Such boundary cases are often difficult to address using automated approaches. In prior work, for example, Breaux et al. discovered heuristics for inferring *implied* rights and obligations from explicitly stated requirements to increase requirements coverage [8]. However, the downside of manual methods is the challenge of scalability: achieving a two-fold increase in the number of documents processed requires considerable effort by a small number of expert analysts. Moreover, automated approaches, including machine learning, rely on large data sets to evaluate performance in cross-validation studies [3]. Overcoming this challenge of scaling manual extraction could lead to new analytics that leverage unprecedently large datasets.

Crowdsourcing and human computation provide a middle ground between manual extraction by a few experts and natural language processing. Crowdsourcing has emerged as a viable approach for leveraging human intelligence – often provided by non-experts – in the context of problems that remain hard to solve with automated methods [33], such as information extraction from noisy images [1], political sentiment analysis [18], and text translation [4, 38]. In crowdsourcing, tasks are typically divided into smaller, more manageable *microtasks*. These microtasks are then distributed to the *crowd* – a large number of independent workers who offer their services using crowdsourcing platforms, such as Amazon Mechanical Turk¹ (MTurk) or CrowdFlower.² Crowdsourcing results are subsequently combined to yield a solution for the larger task. Major challenges in crowdsourcing complex tasks include designing a task workflow that produces high quality results, while keeping per-task costs within budgetary constraints.

In this paper, we report results from three experiments aimed at assessing the potential of crowdsourcing requirements extraction to non-experts. We based our experiments on a recently published method for manually extracting requirements from privacy policies [7] as follows: the first experiment assessed the crowd workers’ ability to apply a coding frame, and measured consensus building across workers; the second experiment assessed the ability of crowd workers to extract complete requirements using multiple

¹ <https://www.mturk.com>

² <https://crowdflower.com>

semantic roles, simultaneously; based on the gained insights from these two experiments, a third experiment assessed a task decomposition approach, in which semantic role labeling was divided into separate, interdependent microtasks to minimize worker effort and increase data quality. We found that our approach yields nearly a 1.5:1 cost decrease and increased requirements coverage compared to manual extraction by trained experts. The contributions of this approach include a validated task decomposition workflow and generalizable metrics for worker performance, which we believe can be used to crowdsource similar requirements extraction tasks.

The remainder of this paper is organized as follows: in Section II we review related work on designing workflows for crowdsourcing complex tasks; in Section III we introduce the manual extraction method for privacy requirements and corresponding economics based on two prior case studies; in Section IV, we present the results of two preliminary crowdsourcing experiments based on the manual extraction method; in Section V, we present our task decomposition approach, experiment and our main results. A discussion follows in Section VI with the conclusion in Section VII.

II. DESIGNING CROWDSOURCING WORKFLOWS

Crowdsourcing has been successfully applied to many different tasks and contexts [13]. Snow et al. [36] showed that crowdsourced annotations from non-experts reach a high level of agreement with expert annotators for different natural language annotation tasks, such as word similarity, word sense disambiguation and textual entailment recognition. We now review related work on data quality and task design.

A. Data Quality

Annotation quality is impacted by crowd worker and task characteristics [2]. Workers are typically non-experts of the task, which may introduce inaccuracies and noise in results. Dishonest workers may try to cheat to obtain payment. Complexity of tasks and instructions of a task can further induce high cognitive demand [15], which may reduce result quality [2]. Thus, crowdsourcing annotation tasks should be understood as a noisy classification process. A number of quality control strategies have been proposed [33] at design or runtime to ensure higher quality results [2]. We now discuss data quality approaches relevant to requirements extraction [2, 33].

Pre-screening and reputation. Workers can be asked pre-screening questions to evaluate their suitability before assigning tasks [32]. A certain reputation level, e.g., 95% accepted submissions, can be used to target high-performing workers, which has been shown to significantly influence result quality for low pay tasks [23].

Review and quality control. After tasks have been submitted by workers, results can be reviewed before accepting them. If a submission is rejected, pay is withheld for that task. Expert review is an expensive approach that limits scalability, because domain experts must review and judge the quality of each submission before accepting them [2]. Multi-level review, on the other hand, leverages a second group of crowd workers to verify data annotations provided by the first group [33]. Automated verification can be used for tasks for which verifying result correctness is easier than its computation [33].

Incentive structure and economic model. Incentive structures can be optimized to obtain higher data quality at lower costs [33]. Provided monetary incentives can impact data quality. Kazai finds that higher paying tasks lead to good worker performance regardless of the qualification [23]. However, she also finds evidence for diminishing returns as workers who are paid low wages report to be happier with their compensation than highly paid workers. Thus, higher pay does not necessarily result in higher quality [2]. Horton et al. describe a labor economics model of crowdsourcing [17]. Their model is based on a reservation wage, i.e., the lowest compensation a worker is willing to accept for a task. In experiments, they determine that the reservation wage is log normally distributed with a median wage of \$1.38/hour.

Redundancy and majority consensus. To improve reliability with noisy data, the same task can be assigned to multiple workers in order to determine output agreement. Output agreement can be measured by majority voting or weighted voting, which takes into account worker performance [33]. A large variety of crowd consensus approaches have been proposed [19], which also resulted in benchmarks to compare their effectiveness [19, 37]. Redundancy, which is needed in all consensus based approaches, may entail higher costs [20].

Identification of low performers. Result quality can be improved (or redundancy reduced) by detecting low-performing or potentially dishonest workers. An effective approach is ground truth seeding [33], in which microtasks for which gold standard data exists are mixed into a worker's assigned task. Failing those tasks can indicate low performance and potentially dishonest intent. At the same time, such verifiable tasks can be used to provide training to workers before they engage in tasks for which no gold standard data exists [24]. Signaling to users that responses will be scrutinized further encourages honest task completion [24]. Statistical filtering can further be employed to determine how far workers deviate from the majority [20, 33]. Once low performers are identified, their results can be rejected and they can be excluded from future assignments, or they can be provided with shepherding and real-time support to improve their performance [2]. Kamar et al. [21] propose a method for routing tasks to workers that optimizes hiring decisions for maximum utility. Based on previously collected task responses as well as knowledge about individual worker performance, they estimate what value additional task submissions would provide to the result.

Defensive task design. Carefully phrasing tasks and associated instructions can reduce ambiguities and thus improve result quality [33]. Kittur et al. [24] note that tasks should be designed so that cheating is no easier than completing the task honestly, i.e., creating believable invalid responses should be as effortful as providing honest responses. Careful arrangement of task order can also increase quality.

While data quality can be measured by accuracy, i.e., the ratio of correct labels to the total number of labels, whether a label is correct is often not objectively verifiable. Hsueh et al. propose practical aspects to be considered by quality metrics [18]: *annotator-level noise* reflects an annotator's accumulated noise level across tasks, whereby noise is defined as a deviation from the majority vote; *item-level ambiguity* reflects that different items are not equally easy to annotate; *inherent*

lexical uncertainty of items can be determined by the uncertainty of an automated classifier for the respective item. Allahbakhsh et al. [2] further note that the subjective nature of quality and the fact that many quality control methods are tailored to specific task domains make it difficult to compare quality metrics.

B. Task Design and Decomposition

Robust task design is an essential aspect of crowdsourcing complex tasks. Instruction length and complexity can increase cognitive demand and thus decrease worker performance [15]. Complex payment schemes also increase cognitive demand, because workers focus on optimizing their payment in addition to completing the task [15]. Thus, task design that channels cognitive demand towards the tasks is potentially more effective. Repeating tasks also facilitates a learning effect for workers which reduces cognitive load and thus potential for error [15]. A related aspect is that the usability of the crowdsourcing interface impacts result quality [2]. Eickhoff and de Vries [14] further find that phrasing tasks in a non-repetitive manner discourages automation on the worker side and yields higher quality results.

Allahbakhsh et al. [2] note that task granularity affects data quality. They distinguish between simple tasks that are self-contained and require low expertise and complex tasks. Complex tasks should be decomposed into simpler microtasks, of which the results can be later consolidated to solve the larger task. Microtasks can be chained to form iterative and parallelized task workflows [2]. TurKit [28] allows to integrate MTurk tasks into program code with the purpose of automating human computation with complex workflows. Kittur et al. propose CrowdForge as a general purpose framework for solving complex and dependent tasks with a map-reduce approach [25]. CrowdForge supports dynamic multi-level partitioning so that workers themselves can decide how to subdivide a task, which then generates new microtasks automatically. Result aggregation can be automated or partially performed by workers. Turkomatic [26] further aims to crowdsource the actual design of a task workflow. Workers decompose tasks on their own while the requester can modify workflows in realtime. Cascade [10] is an automated workflow for creating object taxonomies. Cascade's task decomposition is based on three task primitives: *generate label* (show multiple items, ask to find category), *select best label* (given a single item and multiple labels, select the best fit), and *categorize* (for a single item and multiple categories, vote for each category). The motivation is that these primitive tasks can be completed in 20 seconds and highly parallelized.

Zaidan and Callison-Burch [38] find that an optimized task workflow results in near-professional quality for natural language translations. In their process, statements are first translated redundantly, those translations are then redundantly edited for fluidity by native speakers, and then those edits are again ranked by crowd workers. Based on a quality metric that incorporates the ranking score, as well as sentence and worker features, best fits are selected automatically. Ambati et al. [4] also propose a crowdsourcing workflow for translation tasks in which results are enhanced through multiple levels, including lexical translation, assistive translation, and monolingual synthesis to refine translated sentences. Negri et al. [31] employ task decomposition to build a cross-lingual entailment

corpus with crowdsourcing. Their workflow consists of sentence modification, annotation, and translation tasks. They find that the resulting workflow can be scaled easily for a large set of original sentences.

C. Crowdsourcing and Requirements Engineering

Requirements engineering research is beginning to leverage crowdsourcing in elicitation. Lim and Finkelstein [27] introduce the StakeRare method that employs social networks and collaborative filtering to elicit and prioritize user requirements. The approach was applied to a 30,000 user system with 87 stakeholders and showed improved completeness and accuracy in predicting stakeholder priorities. Caire et al. [9] employ crowdsourcing to elicit visual notations from novices and study their utility. In our work, we investigated task decomposition for requirements extraction in multiple crowdsourcing experiments with the goal of obtaining an optimized and scalable task workflow. Our work centered on the extraction of privacy requirements from privacy policies.

III. MANUAL EXTRACTION METHOD

Our three experiments are based on a manual method for extracting privacy requirements from privacy policies [7]. In the manual method, the analyst first classifies statements using a statement-level coding frame based on action type (e.g., does it refer to collection, use or transfer of personal information). Next, the analyst applies a phrase-level coding frame to identify the information type, recipients or senders of information, and purposes for which information is used. The complete coding frame is presented in Appendix A. In Fig. 1, we present an example policy statement that has been manually coded in a prior case study [7]: phrases are highlighted corresponding to codes for words indicating modality, transfer, datum, target, etc. In the complete manual method, the extracted phrases are mapped to logical formula, which are then used to reason about conflicts between permitted and prohibited actions [7]. In this paper, we only treat the extraction problem, and leave crowdsourcing of translation to logic to future work.

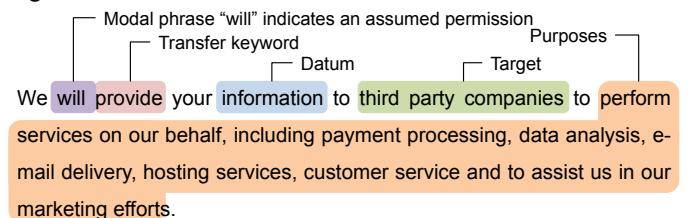


Fig. 1. Example coded policy statement: the statement-level code “Transfer” is mapped to the action verb “provide,” the phrase-level codes identify the modal phrase, datum, target and list of purposes.

There are multiple factors to consider when evaluating how well crowdsourcing can contribute to scaling this task, including the time to extract a requirement, the cost of extraction and the proportion of extracted statements to the size of the policies. In our prior study [7], our analysts spent between 1.4 and 1.8 minutes per policy statement coding the policy text. The cost to code the Zynga privacy policy, which consisted of 189 statements, at a pay rate of \$15 per hour would be \$86.25 or about \$0.46 per statement. To assess validity, we use the inter-rater reliability Kappa statistic, which

requires multiple analysts and which at least doubles the cost to code the Zynga policy to \$172.50. This cost and effort includes the time to read the policy, identify relevant statements, and assign the corresponding codes; it does not include the time to translate the coded phrases into logic or a requirements specification language. In addition, we found that the analysts coded only 28-49% of the policy statements. Many uncoded statements reside in specific sections of the policies that can be eliminated to reduce the cost of the manual effort as well.

In the next section, we describe two pilot experiments and our results to adapt this method to the MTurk crowdsourcing platform. In the third experiment, we discuss how the pilot studies and cost of the manual approach factored into our task decomposition design choices.

IV. PRELIMINARY EXPERIMENTS

The task designs for the first two experiments E1 and E2 aimed at answering two research questions: how well do crowd workers apply the statement-level and phrase-level coding frames, and how many crowd workers are needed to reach consensus? We now discuss the two designs and results.

A. Sentence Classification and Consensus Building

Experiment E1 evaluates the crowd workers' ability to apply the sentence-level coding frame (see Appendix A) and to measure how many crowd workers are needed to reach consensus. For this task, we created a stratified sample of nine sentences selected from our prior case study [7], which we use as a gold standard to evaluate the crowd worker results. We selected sentences along the following strata: *single-coded*, in which only a single sentence-level code applies; *dual-coded*, in which exactly two sentence-level codes apply; and *none*, wherein no codes apply. We chose this strata to ensure coverage across the coding frame and assess how well workers respond to statements with multiple codes. The selected sentences, prefixed by the expected codes in bold, are as follows:

1. **Collect:** "We may collect or receive information from other sources including (i) other Zynga users who choose to upload their email contacts; and (ii) third party information providers."
2. **Consent, Transfer:** "We do not actively share personal information with third party advertisers for their direct marketing purposes unless you give Us your consent."
3. **Retain:** "Zynga stores information about site visitors and players on servers located in the United States."
4. **Collect, Retain:** "We receive and store the information you provide, including your telephone number, when you sign up to have SMS notifications sent directly to your mobile phone."
5. **Transfer:** "To properly credit user accounts and to prevent fraud, a unique identifier, in some cases your user ID number, will be shared with the offer wall provider."³
6. **Use:** "This information will be used to supplement your profile – primarily to help you and your friends connect."

³ An offer wall is a web page or screen with multiple offers aimed at providing users more choices for clicking on ads.

7. **Collect, Use:** "The information collected may be used to offer you targeted ad-selection and delivery in order to personalize your user experience by ensuring that advertisements for products and services you see will appeal to you, a practice known as behavioral advertising, and to undertake web analytics (i.e. to analyze traffic and other end user activity to improve your experience)."
8. **None:** "Zynga implements reasonable security measures to protect the security of your information both online and offline, and We are committed to the protection of customer information."
9. **None:** "Zynga games or their purchase pages may display an 'offer wall' that is hosted by an offer wall provider."

Figure 2 shows the task interface for E1, in which the variable \${text} would be replaced by one of the statements above. The instructions are simple with no examples. Response options (use, transfer, etc.) were always presented in random order with "none of the above" always appearing last. Workers may select as many categories as desired, but they must provide phrases to justify their answers.

Instructions: What action(s) does this statement describe? Check one or more boxes next to those actions and enter any words or phrases from the statement that indicate why you selected the action.

Statement: \${text}

Use - an act to use personal information for a particular purpose
 Use keywords:

Transfer - an act to transfer or share personal information with another party
 Transfer keywords:

Retain - an act to retain or store personal information
 Retain keywords:

Consent - an act by a party to consent to, or control the use of, their personal information
 Consent keywords:

Collect - an act to collect personal information from another party
 Collect keywords:

None of the above

Submit Query

Fig. 2. User interface to the sentence classification task; the variable \${text} is replaced by one of nine sentences; workers are asked to provide matching keywords from the text to justify their responses.

For E1, we recruited US residents as workers on MTurk, who had at least a 95% approval rating for over 5,000 tasks. We paid workers \$0.15 per sentence classification task and allowed up to 20 minutes to complete the task. Results were accepted or rejected within 24 hours. On average, workers required 66 seconds to complete a single sentence classification that resulted in an average hourly rate of \$8.18.

In response to our request, we solicited 50 workers per task that resulted in a total of 76 distinct workers participating who classified between 1-9 sentences each to yield 448 classifications (i.e., workers are not required to complete all tasks). Table I presents the results as the number of ratings per sentence (S#) by category: cells shaded dark blue represent majority consensus (i.e., $\geq 25/50$ workers assigned the category to this sentence) and cells shaded light orange represent near majority consensus or NMC (i.e., 13-24 workers assigned the category to this sentence). The dark blue-shaded cells match

our expected results with 100% precision and recall. After inspecting the results, we believe NMC is due to implied actions perceived by some workers, e.g., the action of “signing up” for a game account (statement 4) implies that the user will transfer some personal information to the game website.

TABLE I. FREQUENCIES OF CODES PER SENTENCE

| S# | Collect | Consent | Use | Retain | Transfer | None |
|----|---------|---------|-----|--------|----------|------|
| 1 | 48 | 8 | 9 | 19 | 15 | 0 |
| 2 | 8 | 46 | 21 | 7 | 38 | 0 |
| 3 | 12 | 0 | 2 | 49 | 3 | 0 |
| 4 | 38 | 16 | 13 | 44 | 8 | 0 |
| 5 | 7 | 2 | 22 | 6 | 40 | 3 |
| 6 | 13 | 4 | 42 | 12 | 10 | 1 |
| 7 | 36 | 5 | 44 | 10 | 10 | 0 |
| 8 | 7 | 3 | 5 | 19 | 0 | 25 |
| 9 | 5 | 2 | 8 | 1 | 8 | 31 |

Due to space limitations, Fig. 3 only presents plots for the percentage of crowd workers who classified even sentences 2, 4, 6, 8: the y-axis shows the percentage of workers who assigned the category, and x-axis shows the response from the i^{th} worker from the 1st to the 50th in time order. Across all 9 plots, we observe that percentages converge at between 5-15 workers per statement, which indicates that higher worker numbers are not necessary. We used this information to set the number of invited workers to complete experiments E2 and E3.

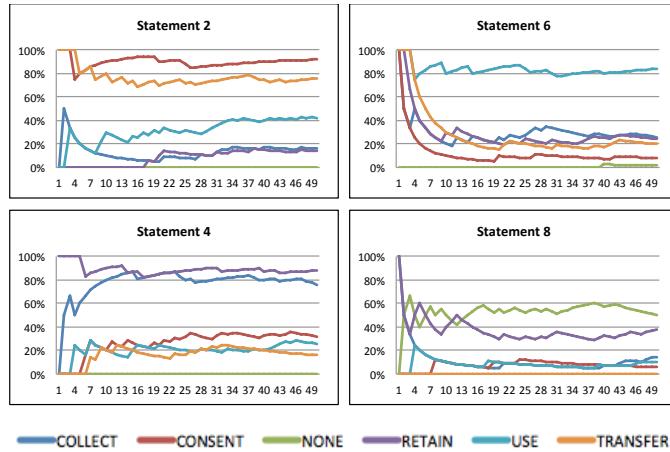


Fig. 3. Percentage of crowd workers who labeled statements by concept; the y-axis is the percentage, and the x-axis is the worker number from the 1st worker to the 50th worker; the proportions stabilize between 5-15 workers.

We evaluated worker performance in E1 by their ability to assign ratings within the majority consensus (i.e., of all their ratings assigned, what proportion match the majority rating). We found for each task that 12/50 workers provided between 40-50% of their ratings within the majority, and 18/50 workers rated with the majority less than 50% of the time. In addition, all workers did not complete all tasks: 16/50 workers attempted only one sentence classification task, whereas 32/50 workers completed all nine tasks. We computed the average number of tasks completed by the worker’s quartile ranking with respect to majority ratings and found that Q1=1.9, Q2=5.0, Q3=6.6, and Q4=7.2, which suggests workers rating in the majority are more likely to complete more sentence classification tasks.

B. Full-frame Extraction

Experiment E2 is a “full-frame” extraction task that combines the sentence-level coding frame with the phrase-level frame described in Appendix A. Workers are shown one sentence and they are asked to answer up to seven questions by completing an online form (see Fig. 4). The questions ask: what is the *modality* (permit, prohibit), what is the *action* (collect, use, transfer, retain), what is the *indicative verb* for this action, what *kind of information* is acted upon (the datum), *to whom* is information transferred (target), *from whom* is information collected (source), and *for what reason* or purpose is the action performed. Alternatively, the worker can indicate that the sentence does not describe any of these actions. We further include clickable instructions that expand to show a working example. The action names listed by the verb textbox are selected from the top three verbs for each action identified in our Zynga case study [7]; thus, “transfer, share, send” were the three most commonly observed verbs indicative of a transfer action. The last four questions appear dynamically based on what action has been selected in the drop-down list. For this task, we selected 18 new sentences from the Zynga case study [7] using the same stratification criteria as in E1: single-coded, double-coded and none.

[Click here to show the instructions.](#)

In the following statement, any pronouns "We" or "Us" refer to the \${company}, and "you" refers to the \${user}.

Statement: \${text}

The above statement prohibits the transfer of...

What verb/noun in the statement indicates this action: (e.g., transfer, share, send, etc.)

What kind of information?

Transferred by whom?

Transferred to whom?

Transferred for what reason?

The above statement does not describe a collection, use, retention or transfer.

Submit Query

Fig. 4. User interface to the full-frame extraction task.

We solicited 15 workers per sentence based on E1 results that show sentence-level classification consensus converges between 5-15 workers. These workers were US residents and they had at least a 95% approval rating for over 5,000 tasks. We paid workers \$0.15 per full-frame extraction and allowed up to 10 minutes for workers to complete the task. Results were accepted or rejected within 48 hours. For E2, we received 38 workers who classified between 1-18 sentences yielding 135 classifications. The average task completion time was 89 seconds with an average hourly rate of \$6.07.

We believe the E2 task was more challenging for the workers than the E1 task. In addition to the longer average time required to complete E2 tasks, we saw that individual workers completed fewer tasks in E2 than in E1. Furthermore, the data quality produced by E2 workers is more varied. Figure 5 presents the number of perfect responses received along the y-

axis and the sentence number along the x-axis, respectively. For each column, we report the average worker score for each sentence, which is computed as a ratio of correct answers per total number of possible responses (e.g., for sentence 1, 92% of the questions asked about this sentence were correctly answered across all 15 respondents; a worker’s response is *perfect* if it has a score of 100%).

From these results, we drew a few observations. First, the darker shaded, blue columns show that more than two-thirds of workers produced perfect scores for the respective sentence. In most of these cases, the average score is above 90%. Second, sentences with medium shaded, red columns exhibit an above average score of 77% or greater, but a relatively low number of perfect responses. This is because the cumulative number of correct answers across all fifteen responses is high, whereas few workers could correctly answer all questions. The lighter shaded, orange columns correspond to sentences for which respondents were equally divided between providing answers or selecting the negative response option, which was the correct answer. This may indicate *acquiescence bias* [30], in which respondents feel compelled to complete the template, despite encountering a sentence that is not clearly classifiable.

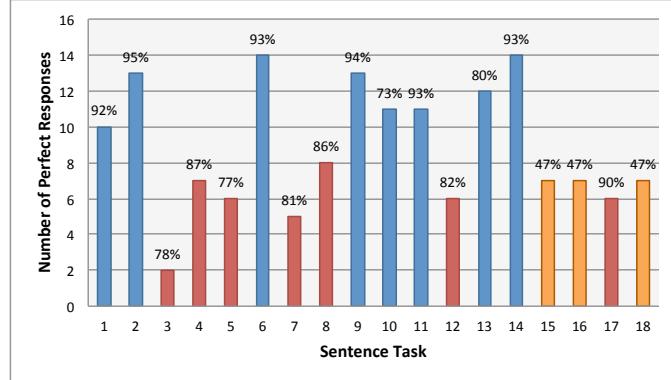


Fig. 5. Number of perfect responses for sentences; percentages correspond to average worker score for each sentence.

While the results from E2 are promising, overall, scaling this full-frame extraction task would be too costly. Recall, the cost for two expert analysts to encode the Zynga policy, which contains 189 statements, is \$172.50. With respect to E2, hiring 15 workers to apply the full-frame extraction to the whole Zynga policy at \$0.15 would cost \$425.25. Therefore, we had to further reduce the number of workers needed and/or increase the quantity of information processed to reduce the overall cost. Therefore, we are inclined to separate the questions in the full-frame extraction into separate tasks, while increasing the quantity of text per task. The idea being that we can recombine the answers in a map-reduce style human computation.

In addition, the E2 results suggest two hypotheses. First, one class of sentences yields perfect responses among two-thirds of respondents (see blue columns in Fig. 5): if we only hire five workers, more than half will produce consensus and perfect responses. Second, a different class of sentences show that only one-third of workers produce perfect responses (see red columns in Fig. 5), but three-fourths of aggregate worker answers in those responses matched consensus with the one-

third who provided perfect responses. These hypotheses informed the experimental design of E3 (see Section V), including the development of three metrics to evaluate which workers produce higher quality responses.

V. TASK DECOMPOSITION FOR EXTRACTION

The results from experiment E2 demonstrate several trade-offs in using untrained crowd workers: (1) combining multiple, heterogeneous microtasks into a larger task increases cognitive demand, which necessitates an increased payment to cover the additional time spent by the worker; and (2) consensus is reached earlier on some microtasks than others. Based on these observations, we designed a task workflow in which the full-frame extraction task from E2 is decomposed by semantic roles into microtasks and distributed to multiple workers. The task decomposition design is driven by assumptions derived from the crowdsourcing literature and our experience as follows:

- A1. Extraction of complete requirements may require the context of multiple sentences or regions of text;
- A2. Once a worker understands a simple task, they can more effectively perform it repeatedly compared to switching between different tasks during the same period [15];
- A3. Certain steps in the extraction process can be performed by automated NLP with acceptable levels of precision and recall [36];
- A4. Certain microtasks depend on the results from other microtasks, thus suggesting natural break points and ordering of microtasks in a task decomposition;
- A5. The financial cost of a task is directly proportional to the task complexity [2, 33] or cognitive demand, thus decomposition should coincide with a smaller cost per microtask, but not necessarily a smaller overall cost.

We designed the task workflow for experiment E3 under these assumptions as follows. First, we chose to provide workers with complete paragraphs, in contrast to E1 and E2 that presented only single sentences. A paragraph provides additional context that may be needed to extract a complete requirement (A1) and we can leverage improved worker performance by having them repeat the same, smaller task for the same duration of 60-90 seconds over larger portions of text (A2). Second, we observed that some microtasks can be partially solved by automated NLP (A3), thus, reducing the workload. Next, some questions need not be answered, if a previous microtask provided a negative result (A4), e.g., we do not need to ask “from whom” information is collected, if the text does not describe collection. These two assumptions A3 and A4 motivate our expectation that we can reduce the cost by skipping portions of policies. The challenge is maintaining validity by ensuring each question is answered by a sufficient number of workers to reach consensus.

Figure 6 presents our task decomposition that follows this line of reasoning. Preliminary microtasks that have not been discussed so far, including downloading privacy policies and extracting paragraphs, are shown and have been realized as semi-automated tasks. The microtask “identify modal verbs,” such as *may*, *will*, *may not*, etc. is automated. Microtask 4 (*identify the action verbs*) is the lead crowdsourcing task: this task is used to evaluate worker performance early based on a list of known action verbs.; if actions are detected by workers,

microtasks 5-7 are created for those paragraphs. Microtask 6 is restricted to paragraphs describing collections or transfers, and actions coded by a majority (>2 raters) are highlighted in this microtask. The results from microtask 5 and 6 are compared with microtask 4 (dotted lines) to assess worker quality.

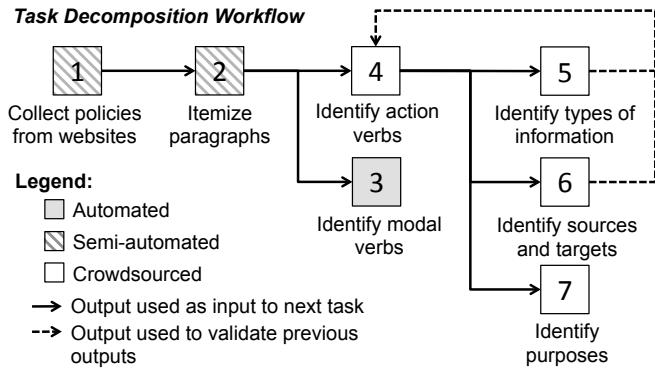


Fig. 6. An overview of the task decomposition workflow.

To effectively support workers in microtask 5, we developed a specialized user interface for coding key phrases (see Fig. 7). In the interface, users select relevant phrases and press “concept” keys to code the phrase with a specific concept. Thus, workers can highlight phrases as they read through the text, as opposed to copying/pasting or typing answers, which can lead to typographic errors that prolong the task and potentially reduce workers’ hourly wages.

We may collect or receive information from other sources including (i) other Zynga users who choose to upload their email contacts; and (ii) third party information providers.



Fig. 7. Example interface used in task decomposition, step 5.

In the experiment E3, we ask workers to complete steps 4-7 of our task workflow for the following five privacy policies: Zynga.com, Rovio.com, Amazon.com, Walmart.com, and Waze.com. We previously studied Zynga's and Waze's privacy policies, which provides a gold standard by which to compare crowd worker results with our manual method. We selected three additional policies to complement this selection: Rovio's policy covers the popular game Angry Birds, which complements Zynga's Farmville game; Amazon and Walmart are both large retail companies, wherein Walmart specializes in brick-and-mortar sales and Amazon specializes in online sales.

In preparation, we downloaded the policies from their websites, itemized the paragraphs and removed sections describing the following content: the introduction, “contact us,” security, U.S. Safe Harbor, policy changes, and California citizen rights. This content is often separated by section or detected with common keywords. In addition, we manually split paragraphs to yield text spans less than 120 words, noting that we preserved some large spans when anaphora referred back to a previous sentence (e.g., when “This information...” depends on a prior sentence to understand *which information* is referred to). We approximate the time to divide these policies to be about 15 minutes per policy, which adds an incidental one-time cost to each workflow instance. Table II summarizes the E3 dataset, including the total policy sections, sentences

count, percent of retained sentences after removing sections, and total number of assignments created per policy. Each assignment is a separate microtask instance in the workflow.

TABLE II. SUMMARY OF POLICIES STUDIED IN EXPERIMENT E3

| Policy | Sections | Sentences | Retained | Assign. |
|---------|----------|-----------|----------|---------|
| Amazon | 16 | 95 | 69.5% | 18 |
| Rovio | 13 | 84 | 83.3% | 18 |
| Walmart | 28 | 169 | 71.0% | 27 |
| Waze | 17 | 135 | 85.7% | 34 |
| Zynga | 36 | 195 | 69.2% | 32 |

We developed a grading system to evaluate worker performance in experiment E3. We cannot use inter-rater reliability to evaluate consensus for phrase-level coding because the non-coded phrases dominate coded phrases and workers generally agree about which phrases not to code, which causes Fleiss' Kappa to converge at 0.99 for all tasks. Thus, we compute a *worker grade* based on the distribution of their ratings among rater agreement as follows:

- Grade 4, if $\geq 50\%$ of their ratings shared by 5 workers
 - Grade 3, if $\geq 50\%$ of their ratings shared by ≥ 4 workers
 - Grade 2, if $\geq 50\%$ of their ratings shared by ≥ 3 workers
 - Grade 1, if $\geq 50\%$ of their ratings shared by ≤ 2 workers

In addition, we compute a *ratings/task measure* that describes the ratio of total ratings to total tasks completed by workers. These two measures balance each other: a worker may submit one “safe” rating per assignment that five workers agree with to earn a Grade 4, whereas a worker who submits more than eight ratings per task may be more likely to yield increased coverage with fewer workers in agreement and earn a Grade 1.

Consider Table III as a sample data set for computing worker grades: the *Workers* column lists a unique index for each of three workers W1-W3; the *Ratings* column lists worker ratings in three ranks R1-R3 where a rating is counted in only one column depending on how many workers agree with the rating (e.g., a rating counted under R3 means exactly three workers assigned the same rating to the same item); the *Rank Proportion* columns list the proportion of worker ratings that fall into one of the three ranks (e.g., W1 produced 60/90 ratings in rank R1, so they have proportion P1 = 66%). Finally, grades are assigned based on the rank proportions, e.g., worker W2 received a grade of 3, since $\geq 50\%$ of their ratings (50/90 under R3) are shared by three or more workers (P3 = .556).

TABLE III. EXAMPLE RANKING AND GRADING COMPUTATION

| Worker | Ranks | | | Rank Proportion | | | Grade |
|--------|-------|----|----|-----------------|-------|-------|-------|
| | R1 | R2 | R3 | P1 | P2 | P3 | |
| W1 | 60 | 20 | 10 | 0.667 | 0.250 | 0.125 | 1 |
| W2 | 15 | 25 | 50 | 0.167 | 0.278 | 0.556 | 3 |
| W3 | 20 | 40 | 30 | 0.250 | 0.444 | 0.333 | 2 |

Finally, we compute a *sanity measure* after microtask 4 to check worker performance against our predicted minimum level of performance. From our prior case study [7], we compiled a list of the 11 most frequently coded action verbs and we use these verbs in three tenses (past, present, present continuous tense) to predict the classification of actions in each paragraph at microtask 4. For each worker who completed this microtask, we compute the *sanity* measure as the ratio of known actions identified by the worker to known actions predicted in the paragraph. Together, these three measures help

us identify workers that contribute different kinds of quality to the results, which we discuss in Section V-A.

A. Results of Task Decomposition Experiment

Table IV presents summary statistics per policy for the four microtasks in experiment E3. We recruited workers who are US residents and who had at least a 97% approval rating for over 5,000 tasks. We paid workers \$0.10 for microtasks 4-5, and we paid \$0.08 for microtasks 6-7, due to their lower complexity. For all microtasks, workers were allowed up to 10 minutes to complete each task. Results were accepted or rejected within 48 hours. For all microtasks, we rejected worker results only if either: (1) the worker produced no codes and 4/5 other workers submitted at least one code for the assignment, demonstrating viable answers missed by the worker; or (2) the worker misunderstood the instructions and coded the wrong microtask (e.g., for information types, they coded actions). The second condition occurred once in microtask 5 and 6, each time with a different worker. When a result was rejected, it was sent back to the platform and another worker was permitted to complete the assignment.

TABLE IV. SUMMARY OF TASKS IN EXPERIMENT E3

| By Policy | Total Assigns. | Total Rejects | Avg. Time | Avg. Pay Rate |
|--------------------|----------------|---------------|-----------|---------------|
| Amazon | 385 | 25 | 63.8s | \$5.18 |
| Rovio | 381 | 21 | 53.8s | \$6.10 |
| Walmart | 571 | 31 | 49.3s | \$6.59 |
| Waze | 732 | 52 | 50.3s | \$6.54 |
| Zynga | 649 | 9 | 63.8s | \$6.18 |
| By microtask | Total Assigns. | Total Rejects | Avg. Time | Avg. Pay Rate |
| 4: Actions | 655 | 10 | 54.0s | \$6.70 |
| 5: Information | 715 | 70 | 50.4s | \$7.25 |
| 6: Sources/Targets | 703 | 58 | 58.6s | \$5.04 |
| 7: Purposes | 645 | 0 | 52.8s | \$5.49 |

Figure 8 presents the total ratings for the six highest-yield workers who produced the most action ratings on the Zynga policy. The lowest grade worker V1, produced the most ratings (i.e., >75) unreported by any other worker. The highest grade workers V3-V6 produced fewer ratings, but generally produced ratings that many other workers agreed with. Across the entire data set, the average ratings per task for grades 1 through 4 were, respectively: 8.01, 7.74, 5.24, and 4.02.

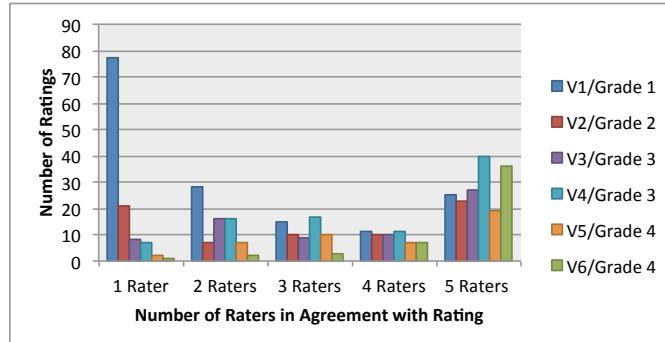


Fig. 8. The six workers with the highest number of attempted assignments for the Zynga policy; workers with more ratings/assignment typically have lower grades, because they find phrases coded by fewer workers.

In the task decomposition, we use a list of known actions from our prior case study [7] to evaluate worker performance.

Table V presents the overall results for predicted responses: for each *Policy*, we show the number of verbs predicted as a collection (C), use (U), transfer (T), retention (R), transaction (collect or transfer, CT) or access (CUT); the number of verbs predicted (*Pred.*), the *Total* number of verbs identified by workers, and the *Ratio* of predicted verbs to total verbs. For the remaining unpredicted verbs, we rely on worker consensus to determine the verbs' classifications.

TABLE V. INFORMATION ACTIONS FILTERED BY KNOWN ACTION LIST

| Policy | Verbs matching code priors | | | | | | <i>Pred.</i> | <i>Total</i> | <i>Ratio</i> |
|---------|----------------------------|----|----|----|----|-----|--------------|--------------|--------------|
| | C | U | T | R | CT | CUT | | | |
| Amazon | 22 | 9 | 9 | 12 | 14 | 8 | 74 | 174 | 0.425 |
| Rovio | 11 | 23 | 9 | 8 | 5 | 5 | 61 | 149 | 0.409 |
| Walmart | 21 | 22 | 15 | 5 | 17 | 4 | 84 | 272 | 0.309 |
| Waze | 3 | 16 | 6 | 4 | 12 | 4 | 45 | 175 | 0.257 |
| Zynga | 11 | 36 | 25 | 2 | 15 | 2 | 95 | 304 | 0.313 |

Based on the predicted verbs, Table VI presents the top six workers who completed the most HITs across all five policies: the table shows a unique *Worker ID*, followed by the number of *Policies* they coded, the total number of *tasks*, the total number of *Coded Items* and *Total Expected* items from the known action list, their *Coded/Expected* ratio, number of *Ratings/HIT* and their *Avg. Grade*, which is the average worker grade across all of their coded items. Notably, the workers with the highest Coded/Expected ratios (V7, V8, and V10) have higher Ratings/HIT and relatively lower grades. These workers not only identify known verbs, but they also identify verbs that few other workers identify. Contrast this behavior with workers V4, V6 and V9, who find only half (or less) of the known verbs, yield few Ratings per HIT, and generally code items that other workers identify (i.e., they have above average grades).

TABLE VI. MOST PRODUCTIVE WORKERS EVALUATED AGAINST VERBS ON THE KNOWN ACTION LIST

| Worker ID | Policies | Tasks | Coded Items | Total Expect. | Coded/Expect. | Ratings/HIT | Avg. Grade |
|-----------|----------|-------|-------------|---------------|---------------|-------------|------------|
| V4 | 5 | 109 | 228 | 438 | 0.521 | 4.1 | 3.0 |
| V6 | 1 | 98 | 241 | 425 | 0.567 | 4.2 | 3.2 |
| V7 | 5 | 73 | 248 | 297 | 0.835 | 12.8 | 1.3 |
| V8 | 3 | 55 | 186 | 236 | 0.788 | 10.0 | 1.8 |
| V9 | 1 | 50 | 70 | 222 | 0.315 | 2.2 | 4.0 |
| V10 | 2 | 35 | 97 | 134 | 0.724 | 9.8 | 1.8 |

B. Comparative Evaluation of E3 and RE'13 Case Study

We compared worker results for the Zynga policy with our RE'13 study results [7] to determine whether task decomposition produced the desired data quality for extracting privacy requirements. Figure 9 presents precision and recall for each of the four microtasks. Notably, precision and recall are inversely proportional, and precision increases with the number of raters in agreement. While all the tasks largely track the same curvature, the recall for identifying actors (i.e., sources or targets) falls sharply by comparison to actions and information types. The hourly rate for the actor identification task was also much lower by comparison. While this task may be perceivably easier given the consistent use of pronouns to describe actors, there are a significant number of such entities in the text. Thus, worker motivation may be decreasing over time which yields fewer correct answers.

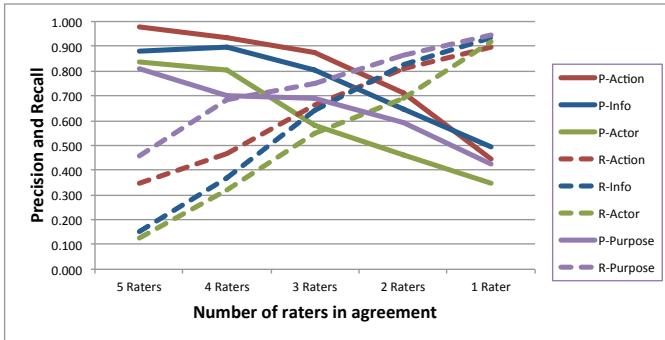


Fig. 9. Precision and Recall comparing the Crowd Worker result with our RE'13 case study result: Precision (P) and Recall (R) are inversely proportional with the number of raters who agree with the ratings.

In addition, we manually inspected the data at the 2-Rater agreement level and found that “false positives” (FP) include implied information actions and types that had previously been missed by the expert analysts in our case study, because the case study was narrowly focused on explicit statements of collection, use and transfer. For actions and 2 raters (recall = 0.811), we found 34/48 FP verbs that could have been reasonably included in our previous case study. This includes verbs, such as *provide*, *deliver*, *notify* and *record* in reference to a kind of information. Among these verbs, 11 described implied information uses and transfers, including *signing up* (for an account), *inviting* (friends) and *remembering* (your last visit). For information types (recall = 0.823), we found 99/119 FP phrases that could be reasonably included in the case study. This includes technology devices, such as mobile devices and browsers, that contain personal information. In experiment E3, however, we instructed workers to report actions performed on any information. Thus, we believe the crowdsourcing result yields a desired increase in extraction coverage.

Table VII presents the total cost for distributing the decomposed task to MTurk for the five policies: at the time of this experiment, the four microtasks cost a combined \$0.36 and were each distributed to 5 workers to yield a \$1.80 unit-cost per task for 5 raters; Amazon charges 10% in *MTurk Fees*, which is added to the *Total Cost*; the last column shows the total *Worker Compensation* that a worker receives for completing all assignments, which we believe motivates workers to participate.

Recall from Section III that the cost for two trained analysts to code the Zynga policy is \$172.50. In addition to the MTurk total cost, there is a one-time cost for sub-dividing the policies (approx. \$5). Based on our task decomposition, we reduced the manual cost by 60% for the Zynga policy and with 2-rater agreement we increased action coverage by 16% with a false positive rate of 3.6%. Based on our average cost of \$0.46 per statement coded by a single trained analyst, and a combined 522 statements for the five policies in our study, the projected cost to have two trained experts code the same five policies would be \$480.24. The task decomposition approach only cost \$305.42, which is a scaling improvement of 1.5:1. As we discuss in Section VI, we discovered additional prospects for improving this scaling factor in the future.

TABLE VII. COST TO CROWDSOURCE THE TASK DECOMPOSITION

| Policy | Tasks* | MTurk Fees | Total Cost | Worker Comp. |
|---------|--------|------------|------------|--------------|
| Amazon | 18 | \$3.24 | \$40.64 | \$6.48 |
| Rovio | 18 | \$3.24 | \$40.64 | \$6.48 |
| Walmart | 27 | \$4.86 | \$58.46 | \$9.72 |
| Waze | 34 | \$6.12 | \$72.32 | \$12.24 |
| Zynga | 32 | \$5.76 | \$68.36 | \$11.52 |

* Assumed a \$1.80 unit-cost per task for 5 raters.

VI. DISCUSSION AND SUMMARY

We now discuss our results from the decomposed task workflow and potential for future work. The three experiments show complimentary results: experiment E1 demonstrates the feasibility of sentence-level coding and the 5-15 worker threshold at which worker agreement stabilizes; in experiment E2, we tasked workers to apply both sentence- and phrase-level codes on a per statement-basis, wherein we discovered that the aggregate of workers’ *partial responses* scored better for most sentences than what workers produced as complete responses; and in experiment E3, we leveraged this insight to create a task decomposition that uses fewer workers and simpler phrase-level coding tasks on larger portions of text to yield an acceptable aggregate response at a reduced overall cost.

Our results show that for a 1.5:1 cost reduction, we can use crowdsourcing to scale requirements extraction for privacy policies. This ability to scale could enable more efficient data flow research in larger data ecosystems [7], as well as comparative studies that contrast practices across business types (e.g., advertisers, social networking sites, etc.) In addition, while trained analysts may fatigue due to the rote task of coding large policies, the task decomposition approach may avoid this fatigue by leveraging a larger population of coders over much smaller tasks (2-3 sentences at a time).

We envision augmenting our crowdsourcing approach with advanced NLP-based techniques that leverage the worker-supplied data. This includes dependency parsers that can help identify actors in the source and target microtask, as well as relevant verbs covering worker-supplied information types. To improve these automated techniques, we plan to build an information type ontology based on these results after conducting additional worker processing to identify synonyms and subsumption relationships. Finally, for machine-learning based NLP, we believe our crowdsourced approach may produce the large training sets needed to train classifiers.

The results show that requirements analysts and researchers can scale requirements extraction using task decomposition and untrained crowd workers, wherein a worker is *untrained* if they are untested against a training regime prior to performing the task. With appropriate pilot studies, we discovered the minimum number of workers, work quality metrics, task size and payment amount for this kind of work (coding text). We envision that other requirements acquisition or analysis tasks will require similar pilot testing to arrive at respective task-specific numbers and can benefit from our approach.

ACKNOWLEDGMENT

We thank Hanan Hibshi, Sudarshan Wadkar, and Anastasia Timoshenko for their helpful feedback. This work was supported by NSF Award #1330596.

REFERENCES

- [1] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, M. Blum. “reCAPTCHA: Human-Based Character Recognition via Web Security Measures.” *Science* 321 (5895): 1465–68, 2008.
- [2] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H.R. Motahari-Nezhad, E. Bertino, S. Dustdar. “Quality Control in Crowdsourcing Systems: Issues and Directions.” *IEEE Int. Comp.* 17 (2): 76–81, 2013.
- [3] E. Alpaydin, *Introduction to Machine Learning*, MIT Press, 2009.
- [4] V. Ambati, S. Vogel, J. Carbonell. “Collaborative Workflow for Crowdsourcing Translation.” *ACM Conf. Comp. Sup. Coop. Work*, pp. 1191–94, 2012.
- [5] T.D. Breaux, *Legal Requirements Acquisition for the Specification of Legally Compliant Information Systems*. Ph.D. Thesis, N.C. State Univ., Apr. 2009
- [6] T.D. Breaux, A.I. Antón, J. Doyle. “Semantic Parameterization: A Process for Modeling Domain Descriptions.” *ACM Trans. Soft. Engr. Method.*, 18(2): 5, Nov. 2008
- [7] T.D. Breaux, H. Hibshi, A. Rao. “Eddy, A Formal Language for Specifying and Analyzing Data Flow Specifications for Conflicting Privacy Requirements.” To Appear: *Req'ts Engr. J.*, 2014.
- [8] T.D. Breaux, M.W. Vail, A.I. Antón: “Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations”, *IEEE Int'l Req'ts Engr. Conf.*, pp. 46-55, 2006.
- [9] P. Caire, N. Genon, P. Heymans, D.L. Moody. “Visual notation design 2.0: Towards user comprehensible requirements engineering notations,” *IEEE Int'l Req'ts Engr. Conf.*, pp. 115-224, 2013.
- [10] L. B. Chilton, G. Little, D. Edge, D. S. Weld, J. A. Landay. “Cascade: Crowdsourcing Taxonomy Creation.” *Conf. Human Factors in Comp. Sys.*, pp. 1999–2008, 2013, ACM.
- [11] J. Cleland-Huang, A. Czauderna, M. Gibiec, J. Emenecker “A machine learning approach for tracing regulatory codes to product specific requirements,” *IEEE Int'l Conf. Soft. Engr.*, pp. 155-164, 2010.
- [12] J. Corbin, A. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, SAGE Pubs., 2007.
- [13] A. Doan, R. Ramakrishnan, A. Y. Halevy. “Crowdsourcing Systems on the World-Wide Web.” *Commun. ACM* 54 (4): 86–96, 2011.
- [14] C. Eickhoff, A. P. de Vries. “Increasing Cheat Robustness of Crowdsourcing Tasks.” *Information Retrieval* 16 (2): 121–37, 2013.
- [15] A. Finnerty, P. Kucherbaev, S. Tranquillini, G. Convertino. “Keep It Simple: Reward and Task Design in Crowdsourcing.” *Biannual Conf. Italian Chapter of SIGCHI*, 2014, ACM.
- [16] D.G. Gordon, T.D. Breaux, “Assessing regulatory change through legal requirements coverage modeling.” *IEEE Int'l Req'ts Engr. Conf.*, pp. 145-154, 2013.
- [17] J. J. Horton, L. B. Chilton. “The Labor Economics of Paid Crowdsourcing.” *ACM Conf. Electronic Commerce*, pp. 209–18, 2010.
- [18] P.-Y. Hsueh, P. Melville, V. Sindhwani. “Data Quality from Crowdsourcing: A Study of Annotation Selection Criteria.” *NAACL HLT 2009 W'shp Active Learning for NLP*, pp. 27–35. 2009.
- [19] N. Q. V. Hung, N. T. Tam, L. N. Tran, K. Aberer. “An Evaluation of Aggregation Techniques in Crowdsourcing.” *Web Info. Sys. Engr.*, 2013, Springer Berlin Heidelberg.
- [20] P. G. Ipeirotis, F. Provost, J. Wang. “Quality Management on Amazon Mechanical Turk.” *SIGKDD Workshop on Human Computation*, pp. 64–67. 2010, ACM.
- [21] E. Kamar, S. Hacker, E. Horvitz. “Combining Human and Machine Intelligence in Large-Scale Crowdsourcing.” *Conf. Auto. Agents & Multiagent Sys.*, pp. 467–74, 2012.
- [22] G. Kaur, “Analyzing Email Archives to Better Understand Legal Requirements,” *W'shp Req'ts Engr. & Law*, 2009, pp. 21-26.
- [23] G. Kazai. “An Exploration of the Influence That Task Parameters Have on the Performance of Crowds.” *CrowdConf 2010*.
- [24] A. Kittur, E. H. Chi, B. Suh. “Crowdsourcing User Studies with Mechanical Turk.” *SIGCHI Conf. Human Factors in Comp. Sys.*, pp. 453–56, 2008, ACM.
- [25] A. Kittur, B. Smus, S. Khamkar, R. E. Kraut. “CrowdForge: Crowdsourcing Complex Work.” *ACM Symp. User Interface Software and Technology*, pp. 43–52, 2011, ACM.
- [26] A. Kulkarni, M. Can, B. Hartmann. “Collaboratively Crowdsourcing Workflows with Turkomatic.” *Conf. Comp. Sup. Coop. Work*, pp. 1003–12, 2012, ACM.
- [27] S. L. Lim, A. Finkelstein. “StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation,” *IEEE Trans. Soft. Engr.*, 38(3): 707-735, 2012.
- [28] G. Little, L. B. Chilton, M. Goldman, R. C. Miller. “TurKit: Human Computation Algorithms on Mechanical Turk.” *ACM Symp. UI Soft. & Tech.*, pp. 57–66, 2010.
- [29] A.K. Massey, J. Eisenstein, A. Anton. “Automated text mining for requirements analysis of policy documents.” *IEEE Int'l Req'ts Engr. Conf.*, pp. 4-13, 2013.
- [30] S. Messick, D.N. Jackson. “Acquiescence and the factorial interpretation of the MMPI.” *Psych. B.*, 58(4): 299-304, 1961.
- [31] M. Negri, L. Bentivogli, Y. Mehdad, D. Giampiccolo, A. Marchetti. “Divide and Conquer: Crowdsourcing the Creation of Cross-Lingual Textual Entailment Corpora.” *Conf. Emp. Meths. in NLP*, pp. 670–79, 2011, ACL.
- [32] G. Paolacci, J. Chandler, P. G. Ipeirotis. “Running Experiments on Amazon Mechanical Turk.” *Judge. & Dec. Making* 5 (5): 411–19, 2010.
- [33] A. J. Quinn, B. B. Bederson. “Human Computation: A Survey and Taxonomy of a Growing Field.” *Conf. Human Factors in Comp. Sys.*, pp. 1403–12, 2011, ACM.
- [34] K. Ryan, “The Role of Natural Language in Requirements Engineering,” *Int'l Symp. Req'ts Engr.*, 1993, pp. 240-242.
- [35] J. Saldana, *The Coding Manual for Qualitative Researchers*, SAGE Pubs, 2012.
- [36] R. Snow, B. O'Connor, D. Jurafsky, A. Y. Ng.. “Cheap and Fast—but Is It Good?: Evaluating Non-Expert Annotations for Natural Language Tasks.” *Conf. Emp. Meths. in NLP*, pp. 254–63, 2008, ACL.
- [37] A. Sheshadri, M. Lease. “SQUARE: A Benchmark for Research on Computing Crowd Consensus.” *AAAI Conf. Human Comp. & Crowdsourcing*, 2013, AAAI.
- [38] O. Zaidan, C. Callison-Burch. “Crowdsourcing Translation: Professional Quality from Non-Professionals.” *ACL*, pp. 1220–29, 2011.

APPENDIX A.

A. Statement-level Coding Frame

The statement-level coding frame was established in a case study to extract data flow requirements from privacy policies [7]. In this study, relevant statements in a policy were classified as one of the following (irrelevant statements were not coded):

- Collect* – any act by a first party to access, collect, obtain, receive or acquire data from another party;
- Consent* – any act by a party to consent to, or control the use of, their personal information;
- Use* – any act by a first party to use data in any way for their own purpose;
- Retain* – any act by a first part to retain data for a particular period of time or in a particular location; and
- Transfer* – any act by a first party to transfer, move, send or relocate data to another party.

B. Phrase-level Coding Frame

After the statement-level code is assigned, the analyst would then apply a phrase-level coding frame corresponding to semantic roles [6]. These roles consist of the following codes:

- Modality* – whether the action is a permission, obligation or prohibition;
- Subject* – the actor who performs the action on the datum;
- Datum* – the information on which the action is performed;
- Purpose* – the purpose for which the action is performed; this role applies to any action, but is especially necessary to describe use of the datum.
- Source* – the source from which information is collected;
- Target* – for transfer actions, the recipient to whom the information is transferred.

Discovering Affect-Laden Requirements to Achieve System Acceptance

Alistair Sutcliffe, Paul Rayson, Christopher N. Bull, and Pete Sawyer

Lancaster University

Lancaster, UK

{a.g.sutcliffe, p.rayson, c.bull, p.sawyer}@lancaster.ac.uk

Abstract—Novel envisioned systems face the risk of rejection by their target user community and the requirements engineer must be sensitive to the factors that will determine acceptance or rejection. Conventionally, technology acceptance is determined by perceived usefulness and ease-of-use, but in some domains other factors play an important role. In healthcare systems, particularly, ethical and emotional factors can be crucial. In this paper we describe an approach to requirements discovery that we developed for such systems. We describe how we have applied our approach to a novel system to passively monitor users for signs of cognitive decline consistent with the onset of dementia. A key challenge was eliciting users' reactions to emotionally charged events never before experienced by them at first hand. Our goal was to understand the range of users' emotional responses and their values and motivations, and from these formulate requirements that would maximise the likelihood of acceptance of the system. The problem was heightened by the fact that the key stakeholders were elderly people who represent a poorly studied user constituency. We discuss the elicitation and analysis methodologies used, and our experience with tool support. We conclude by reflecting on the affect issues for RE and for technology acceptance.

Index Terms—Requirements engineering, Affect-laden requirements, Emotional requirements.

I. INTRODUCTION

Novel envisioned systems face the risk of rejection by their target user community, and the requirements engineer must be sensitive to the factors that will determine acceptance or rejection. Conventionally, technology acceptance is thought to be determined by perceived usefulness and ease-of-use [9], but in some domains, other factors play an important role. In healthcare systems, particularly, ethical and emotional factors can be crucial. In this paper we describe an approach to requirements discovery that we have developed for such systems.

In recent years, opportunities have emerged to detect the symptoms of a range of medical conditions by applying computer-based sensing techniques in novel user and social contexts [1]. The work described in this paper took place as part of an investigation into the detection of signs of cognitive decline that may presage the onset of dementia. In addition to the novel technology such a system demands, its development poses interesting requirements engineering (RE) challenges that

are, we believe, typical of an emerging class of software applications intended to contribute to health and well-being.

The December 2013 G8 Summit [35] reflected a growing world-wide concern about dementia. In the UK, only 50% of people with dementia ever receive a diagnosis and for those who do, it is often too late for optimal treatment and support. Early diagnosis facilitates interventions which can significantly improve the long-term outcome of (e.g.) Alzheimer's Disease, and treat other disorders such as depression, anxiety and underlying medical conditions, which can lead to reversible memory dysfunction [2].

To increase the number of referrals, it is clear that novel approaches are needed to stimulate greater awareness of cognitive change and the importance of assessment. To this end, the SAMS project (Software Architecture for Mental health Self-management) is investigating the potential of computer sensing for inferring change in cognitive function. However, the efficacy and acceptability of the SAMS approach depend critically on discovery of the user requirements.

The obstacles to understanding these requirements include not only those that are generic to imagined systems for which few contemporary analogues exist, but also a challenging mix of ethical and emotional factors. In addition, the envisioned non-clinical users are predominantly senior citizens, an age group that has been greatly under-represented in requirements studies and for which (e.g.) established techniques such as the Technology Acceptance Model [9] are suspect. In this paper we describe the mix of techniques and technology that we have used to discover requirements for SAMS.

Two contributions are made by our work. The first is that we highlight some of the **requirements challenges** involved in understanding what will make users accept or reject systems they cannot be obliged to use. The second contribution is the **method** for discovery and analysis of requirements for such systems that we have developed and applied in one on-going case study.

The rest of the paper is structured as follows. RE for health and well-being applications, RE for affect, and text mining tools for RE are reviewed in section II. Section III provides a brief description of the project that forms the context of our work. Section IV introduces the method we have developed for the discovery and analysis of a system, the use of which is entirely at the discretion of its target users, who may be

influenced by a mix of concerns. Section V describes in detail how we have applied our method and the results we obtained, both in terms of the requirements knowledge gained and the performance of the support tool used. Section VI discusses the results and section VII concludes the paper.

II. RELATED WORK

In the health informatics literature, requirements are frequently neglected [3], although design rationale in the Goals-Questions-Results method [22] has been used as a shared representation. Jorgensen and Bossen [16] proposed executable use cases coupled to validation via informal animated diagrams for RE in pervasive healthcare. Garde and Knaup [13] argued for a grounded theory approach to RE in healthcare domains to deal with the complexity of the domain and socio-political issues; while participatory design [23] and ethnographic approaches [15] have been applied successfully in healthcare. Cysneiros [8] reviewed a variety of requirements elicitation techniques which could be applied to healthcare, suggesting that technique combination might be more effective. Technique combination (scenarios, prototypes and linguistic corpus analysis) was successfully applied to a healthcare decision support system [33].

A taxonomy of affect-related issues articulated as user-oriented values, motivations and emotions was described by Thew and Sutcliffe [34], with limited explanation of possible implications for values and motivations in the requirements process or for high-level user goals. Ramos and Berry [25] have provided evidence for the impact of emotions on system acceptance. A more detailed taxonomy of social and political RE issues with guidelines for recognising affective reactions among stakeholders was proposed by Ramos et al. [25], [26], who applied their approach in analysing requirements for ERP applications. Callele [6] has applied the concept of emotion in requirements for games applications; however, affect generally has received little attention in RE.

We employ text mining to assist our discovery of affect, where text mining here serves as a convenient umbrella term for any technique classifiable as natural language processing (NLP) or information retrieval (IR). Text mining has attracted significant interest from the RE community and has been applied to a range of requirements problems. Text mining techniques have proven most useful for processing large volumes of text (requirements statements, elicitation transcripts, etc.) where the effort required of a human to process the text is high, and likely to lead to errors due to attention slips and lapses. In such problems, text-mining techniques' inevitably imperfect performance can be most favourably traded off against reductions in effort and human error.

Automatic link generation in requirements tracing [7], [14] is a good example of such a problem. Here, being able to automatically infer *derived from* relationships by the application (e.g.) TF-IDF is attractive, provided the omission of a minority of genuine trace relationships that the tool will fail to identify can be tolerated [4].

The automatic generation of (e.g. graphical) models by the application of text-mining techniques to textual requirement statements has also generated interest (e.g. [24]). However, the comparative lack of success here illustrates one of the key challenges for text mining in RE. The automatic generation of models requires of NL requirement statements a degree of completeness, precision and absence of tacit information that humans simply don't need [29]. By contrast, humans who have sufficient experience and domain knowledge are able to tolerate a relatively high degree of incompleteness, imprecision and tacitness, yet still make a reasonable interpretation of a requirement's intent. There is thus a fundamental mismatch between humans' use of, and need for, economy of expression, and automatic techniques' need for completeness and explicitness.

A consequence of this mismatch is that text that is written for human readers is resistant to the automatic extraction of semantics and pragmatics. This means that the automatic synthesis of requirements from transcripts of elicitation exercises is extremely difficult. Despite this, several text-mining techniques operate successfully at the lexical level to infer *shallow* semantics. Well known examples include Latent Semantic Analysis [10], which has been applied to link generation, and Latent Dirichlet Allocation [5] used in topic modelling.

Corpus linguistics combined with NLP can be used to infer properties of a document by comparison to a large corpus of text whose properties are known *a priori*; this has also found application in RE [27], [30], particularly for abstraction identification [12]. Here, shallow semantics can be inferred from lexical form and context to (e.g.) classify a document focus in terms of semantic categories.

Sentiment analysis [36] has the potential to aid the understanding of affect. However, sentiment analysis focuses on classifying text as expressing positive or negative opinions on a specific topic. This could be useful for (e.g.) crowdsourcing an assessment of the demand for possible new features. Understanding why and how affect influences stakeholder choice is a different problem, however, and the one we address in this paper.

III. SAMS

SAMS' long-term goal is to help increase the proportion of dementia sufferers receiving an early diagnosis. At its core is a set of passive monitors that collect data as a user interacts routinely with their computer. This data is analysed to infer the users' cognitive health against a set of clinical indicators (CIs) representing (e.g.) memory, motor control, use of language, etc. For example, loss of vocabulary is a common symptom of dementia [18], [20], and this may be discoverable by text mining if (e.g.) the user uses e-mail or social networks to keep in touch with kin. If SAMS accumulates evidence consistent with early dementia, it will issue an alert to the user, suggesting they take a follow-up test such as MoCA [21]. Such tests are claimed to have good diagnostic fidelity and should stimulate the user to visit their family doctor for a full clinical assessment.

There are many technical and clinical hurdles to overcome if SAMS is to work, not the least of which is how to interpret the values ascribed to the CIs from the monitored data. However, even if these challenges can be overcome, SAMS will fail if user acceptance is not achieved. Maximising the prospect of user acceptance was the aim of the requirements elicitation and analysis approach described in the next section. We focused particularly on discovery of requirements arising from:

- users' sensitivity to being monitored and the use to which the data was put, and;
- users' reaction to an alert suggesting they take a follow-up test (and by implication that something was wrong with them).

Both of these are affect-laden. In particular, discovery of requirements arising from the second fall under our classification [32] of *unknown unknowns*, since few users will have experienced anything analogous at first hand.

IV. METHODS AND TECHNIQUES

SAMS requirements discovery was initiated with five workshops that were conducted with a total of 24 participants (14 male, 10 female, age range 60-75, median 66), with a median four participants/session plus two facilitators and one to two moderators from the Alzheimer's Society (AS) or the Dementias and Neurodegenerative Diseases Research Network (DeNDRoN).

All workshops were structured in two sessions lasting approximately 1 hour. In the first session the SAMS system aims, major components and operation were explained followed by presentation of eight PowerPoint storyboards illustrating design options for the alert-feedback user interface, such as choice of media (video, text, computer avatars), content (level of detail, social network) and monitoring (periodic feedback, alert only, explicit tests). The second session focused on discussion of privacy issues in monitoring computer use, data sharing and security, ethical considerations, emotional impact of alert messages, users' motivations and likelihood of taking follow-up tests.

Requirements issues raised in the workshops were explored further in 13 interviews following a similar structured approach of explaining the SAMS system, presenting scenarios to illustrate similar design options with discussion on privacy, security and ethical issues. Questions in the interviews also probed users' reactions to different levels of monitoring (e.g. actions, text) and their perceived trade-off between benefits/motivations versus fears/barriers for adopting the system and taking follow-up action after an alert message. Respondents (4 male, 9 female), ranging from 67 to 89 years old (median 72), were all interviewed in their own homes, apart from three sessions carried out in a community centre.

All workshops and interviews were audio-recorded with the participant's consent. Interview participants were also invited to use an audio recorder for one week after the interview date to record any issues which they subsequently thought might have been included in the interview, and their feelings related to any personal experiences of news stories relevant to the

domain, i.e. dementia and Alzheimer's Disease. Two news stories on Alzheimer's Disease (medical progress, personal story of younger patients) taken from the BBC News website were left with the participants if they needed material to prompt reflections.

V. ANALYSIS RESULTS

Analysis of the interviews and workshops are presented in two sections: first a standard requirements analysis treatment of listening to audio recordings and distilling interview notes to produce a list of functional and non-functional requirements. The second section reports more in-depth manual and tool-supported analyses of transcriptions of the interviews and workshops to produce a thematic analysis of requirements issues, stakeholder reactions to designs, emotional feelings about the system, ethical and privacy concerns.

A. Conventional Requirements Analysis

1) Workshop Results

(a) *Reaction to Design Options:* Opinion was never unanimous on any design option. There was no consensus on choice of media (text/video/avatar), although a majority in all workshops favoured provision of more detail and availability of regular reports (content). In addition, most favoured having an icon serving as a visual cue to remind them that the system was running, with a control that temporarily disabled the monitoring. Use of video was favoured in four workshops where participants suggested that self-help (how to cope) and explanatory videos (dementia mitigation treatments) were important motivators for persuading them to take follow-up action. Active monitoring (e.g. quizzes) was favoured by all, but (e.g. card) games were rejected in three of the five workshops. Participants in all workshops suggested that configuration controls for different design options would be welcome.

(b) *Privacy, Ethics and Emotions:* All participants expressed concerns over privacy and security arising from monitoring their computer use. Although they were reluctantly willing to share their data with the researchers for analysis, most participants insisted they should have control over their own data. Sharing data with their close kin/friends had to be under their control and the majority would not share information or the alert with their doctor. The majority in all workshops were willing to allow monitoring of their computer use and e-mail text content, suitably anonymised to protect the identities of other parties to conversations. Most participants expected to experience anxiety and fear if they received an alert message, although they all stated that they would take a follow-up test. Contact with a human expert or carer was cited as an important support, with connections to support groups (e.g. the Alzheimer's Society) as additional sources of information to motivate people to take follow-up tests.

2) Interviews

(a) *Reaction to Design Options:* The interviews produced even less consensus than the workshops for the user interface design requirements. Most respondents (11/13) favoured the

plain text alert message over other media options, although the reminder icon with the disable control (11/13) was a shared requirement with the workshop participants. Active monitoring by a ‘cognitive quiz’ and a weekly diary was favoured by the majority (11/13) although card games were less popular (8).

(b) *Privacy, Ethics and Emotions*: The respondents were even more concerned about privacy and security, possibly because three participants had recently experienced phishing attacks on the Internet. However, only two individuals were not willing to have their e-mail content monitored. Opinions on minimal data sharing and the need to maintain control over their own data were similar to the workshop participants’. The majority of the respondents (11/13) expressed anxiety about being monitored, and expected to experience discomfort, fear and worry when they received an alert message, although all these 11 participants stated they would take the follow-up test: ‘better to know the bad news’ was a common statement. However, ten respondents reported that they could not realistically imagine how they would react in a real-life situation. Five individuals noted that further explanation after the alert message would be vital and all reported that their main motivation for using the system was efficacy: a feeling of being in control by self-management of their health.

3) Requirements Conclusions

Given the diversity of opinion in both the workshops and interviews and inconsistencies between the two modes of requirements capture, prioritising requirements from this analysis was not an easy task. The following requirements list contains issues which supplement the preceding narrative summary of the interviews and workshops. After discussion within the project team, the following conclusions were agreed:

(a) Essential Requirements:

- i. Monitoring computer use and e-mail text, but not other applications
- ii. Active monitors such as a cognitive/general quiz.
- iii. Disable monitoring control always visible
- iv. Simple text alert message.

(b) Desirable Requirements:

- v. Configuration controls to turn off/on the following options (default off in all cases)
 - a. More detailed displays: graphs and chart
 - b. Continuous alert icon
- vi. Active monitors as diaries, weekly quiz and card games
- vii. Video explanation of alert messages and support advice

(c) Additional Requirements- explored but unlikely to be prioritised:

- i. Avatar agents for explaining alerts
- ii. Chatbot agents to gather text via conversations and explanations
- iii. Social media option: closed groups for data sharing and support

(d) Non-functional Requirements:

Privacy and security: controls over any data sharing, encryption and secure transmission of data to university

site, encryption on own PC to mitigate hacking attacks, depersonalised data only for wider research sharing.

Maintainability: installing SAMS on user’s laptop/PC should not disrupt normal computer use.

Performance: SAMS software should not degrade the performance of the user’s machine.

(e) *Emotions and Values*: Several issues which were categorised as values (see [34]) and emotional requirements [25] appeared to have an important bearing on the requirements and design options.

Trust: in the SAMS system, the universities (system authors), healthcare professionals, follow-up test websites and authors thereof.

Motivations: efficacy, desire for self-control, altruism: participation might help research on dementia.

Emotion: anxiety and fear of negative alert messages, uncertainty over personal reaction.

The non-functional requirements, emotions and values emerged to be both critical to SAMS’ acceptance and difficult to get right. These therefore formed the focus of the subsequent thematic analysis.

B. Thematic Analysis

The thematic analysis was performed on the interview recordings and the post-interview recordings, all of which were manually transcribed. The users’ text (without the interviewer’s contributions) comprised a total of 41,000 words. It was analysed to try to find additional insights into what would help or hinder SAMS acceptance, and to do this in a way that provided quantitative evidence we could use to inform SAMS requirements.

The analysis took two modes:

- a data-driven mode where we mined the text and followed where the data took us; and
- a hypothesis-driven mode where we filtered the text to find evidence of ethical, security and emotional concerns.

In both modes, we performed a manual analysis and a supervised semi-automatic analysis using Wmatrix [27].

Wmatrix provides automated inference of properties of the text, integrated within a framework designed for supervised analysis. It uses techniques from corpus-based natural language processing for the shallow semantic annotation of words and phrases; and a hybrid combination of rule-based and probabilistic approaches to assign a part-of-speech label (e.g. noun, verb) and a semantic field label to each word or phrase in the text. The semantic taxonomy (USAS) [28] used has approximately 230 word-sense classes, each represented by a different tag. It is lexicographically based and derived from McArthur’s classification [19] that has been considerably extended and expanded. The semantic tagger aims to identify the coarse-grained contextually correct meaning of a word or phrase. The semantic tagger’s accuracy is around 91% on general written and spoken English.

Wmatrix extracts frequency profiles of words, phrases, grammatical and semantic categories and allows the analyst to compare two or more profiles together using the log-likelihood

(LL) metric. This highlights key words or concepts that are unusually frequent in a text relative to a general reference corpus of spoken English.



Fig 1 Wmatrix semantic cloud of the interviews

Figures 1 and 2 show word ‘clouds’ that are actually formed of the dominant semantic categories in the respondents’ interview responses. The larger the font size, the more terms belonging to the corresponding category dominate. In addition, texts can be queried for occurrences of specific semantic categories, such as emotion or knowledge, to see contextual examples of where those concepts occur in the data. It was for this range of capabilities that we selected Wmatrix for the study. Thus a typical Wmatrix *modus operandi* is to list tag frequencies in rank order, or filter on a subset of tags, list the terms to which these tags correspond and then manually investigate instances of these terms in context, using concordances.

1) Data-driven Analysis

The data-driven analysis was used to explore the main themes of the interviews and the post-interview recordings. A manual discourse function analysis was followed by a frequency-profiling analysis.

Interviews and post-interview recordings were analysed first for discourse function at the paragraph conversational exchange level. A subsequent, more detailed analysis focusing on values, motivations and emotions (VME) expressed in short utterances or phrases, was performed as part of the hypothesis-driven analysis, and is described below.

a) Discourse Function Analysis: The aim of the discourse analysis was to characterise the respondents’ views on issues such as privacy and on the design options illustrated in the scenario mock-ups.

The interviewer’s turns were composed of questions and explanations referencing the scenarios, privacy issues arising from monitoring, and socio-technical aspects of the system. Since the interviewer’s discourse followed a planned script this is not reported in detail. The respondents’ contributions were classified as shown in Table I.



Fig 2 Wmatrix semantic cloud of the post-interviews

TABLE I

RESPONDENT CONTRIBUTIONS

| | |
|-------------------------------------|---|
| Questions of the interviewer | These were to do with <i>monitoring privacy issues, clarification of the scenarios being presented, or other</i> |
| Reaction to scenario Qs | Classified as <i>positive, negative or neutral</i> |
| Reaction to issue Qs | Also classified as <i>positive, negative or neutral</i> |
| Justification | For the responses given |
| Reflection | Classified as <i>general, personal history (dementia experience, kin, etc.) or self</i> |
| Computer experience | Classified as <i>general (novice/expert), use-specific episodes, devices and applications, kinds of activity.</i> |
| Other conversation | |

The more frequent respondent categories were self reflection (10.8%), then other conversation (10.7%), followed by neutral reaction to issue (7.9%), then positive reaction to issue, computer device and question-other, all at 6.5%. Frequencies of other categories ranged from 1-5%. The net valency (positive minus negative reactions, ignoring neutrals) of reaction to the scenarios was +34 with the reaction to monitoring privacy issues slightly less favourable at +28. However, the group-level response masked considerable individual differences, as illustrated in Table II.

The two individuals who asked the most frequent questions (3 & 9) were also the most negative overall; however, the frequency of questions did not correlate with response valency overall. The response valency of issues and scenarios did correlate ($p < .05$, Spearman test), so people were consistent in their judgement. Respondents 3, 4, 5, 9 and 13 appear to be potential non-adopters of SAMS, whereas the rest appear to be moderate to strong potential adopters. These individuals also accounted for most of the privacy value concerns (72%), so invasion of privacy appears to be the main barrier to system acceptance.

The post-interview audio recordings were analysed with the categories listed in Table I, with the addition of further Reflection categories, on: *dementia, medical research on*

dementia, healthcare/National Health Service (NHS) issues, news stories, and personal experiences. The more frequent categories were personal history (17.9%), personal experiences (14.8%), reflections on news stories (9.5%), dementia (8.6%) medical research and healthcare issues both 6.3%. There were no obvious patterns in the data and few valenced reactions to privacy issues or SAMS.

TABLE II
DISTRIBUTION OF RESPONDENT QUESTIONS AND REACTIONS

| Respondent no. | Questions | Net reaction-privacy | Net reaction-scenarios | Overall net valency |
|----------------|-----------|----------------------|------------------------|---------------------|
| 1 | 19 | 1 | 7 | 8 |
| 2 | 14 | -1 | 7 | 7 |
| 3 | 43 | 0 | -1 | -1 |
| 4 | 10 | -1 | -2 | -3 |
| 5 | 13 | 1 | -4 | -4 |
| 6 | 14 | 6 | 5 | 11 |
| 7 | 29 | 6 | 6 | 12 |
| 8 | 9 | 6 | 9 | 15 |
| 9 | 37 | -1 | 0 | -1 |
| 10 | 9 | 3 | 4 | 7 |
| 11 | 16 | 8 | 2 | 10 |
| 12 | 22 | 1 | 5 | 6 |
| 13 | 12 | -1 | -2 | -3 |
| Totals | | +28 | +34 | |

b) *Frequency Profiling*: The purpose of frequency profiling is to identify the most significant (not necessarily the most numerous) terms or concepts within a document. Here, we performed frequency profiling of semantic categories, to try to identify the dominant themes in the interview and post-interview transcripts. Note that in the manual discourse analysis, a tailored set of categories was defined (Table I) that was intended to classify the themes at the paragraph level. Wmatrix applies the USAS tags at the word or multi-word term level. Further, the USAS tags are general-purpose and do not map directly on to the tags in Table I. On the one hand, this makes it hard to use Wmatrix to directly validate the discourse analysis. On the other hand, it allows us to ‘slice and dice’ the transcripts in different but complementary ways.

Wmatrix was first applied to two documents: the consolidated interview transcripts and the consolidated post-interview recordings. As with the manual analysis, clear differences distinguished the two data sets.

These marked differences are illustrated by comparing Figures 1 and 2. Focusing initially on Figure 1, terms related to the solution domain (the tags *information technology and computing* and *Telecommunications*) occur frequently in the text. This suggests that more time in the interviews was spent exploring the design alternatives of SAMS, rather than respondents’ emotional responses (e.g. the tag *worry*) or the problem domain (e.g. the tag *medicines and medical treatment*).

Terms tagged *information technology and computing* (e.g. *computer*, *laptop*, *website*) were the most over-represented category, with a log-likelihood (LL) of 704.74 representing a highly significant degree of over-representation relative to the British National Corpus’s spoken English subset. This compared to an LL of 82.76 for terms tagged *medicines and medical treatment* (e.g. *doctor*, *diagnosed*, *blood-test*), which still occurred significantly more often than predicted by the corpus, but less so than the IT and computing terms.

Note that there is always noise in such an analysis. For example, *pronouns* over-occur significantly, largely because the respondents frequently used first person personal pronouns when prompted to relate (e.g.) their preferences and experiences. It is because of this noise that Wmatrix is designed for supervised use. Clicking on one of the categories gives the concordances of all occurrences of terms tagged with that category, allowing the context of use to be verified. For example, clicking on *information technology and computing* gives direct access to all 215 instances of such words in the interview transcripts, with context, such as:

I like the idea of being told to take an online test. So I quite like that. Just flash up.

To filter out the noise, the individual respondents’ contributions to the interviews were profiled (Figure 3) using the following subset of the most dominant semantic categories, represented by the USAS tags:

| | |
|--------|---------------------------------|
| Y2 | IT and computing |
| Q1.3 | Telecommunications |
| A1.5.1 | Using |
| X2.4 | Investigate, examine |
| B3 | Medicines and medical treatment |
| B2- | Disease |

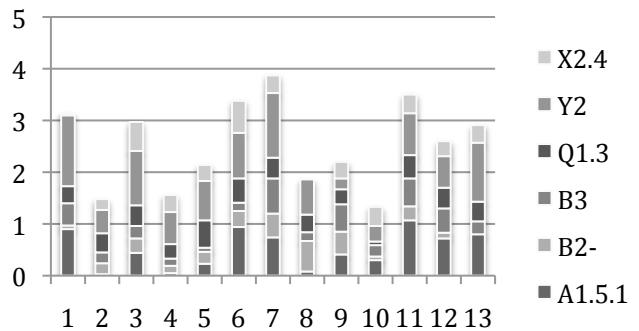


Fig 3 Respondents’ interview contribution profiles

The y-axis represents the relative frequency of occurrence of terms tagged with one of the six tags in the respondents’ interview transcripts. The overall mean for the cumulative relative frequency of the six tags was 2.53. The figure shows that respondents 2, 4, 8 and 10 contributed significantly less relative to the others in discussions on these themes. Respondent 10, for example, mentions e-mail only once, a term that is tagged with *Q1.3 Telecommunications*. This implied indifference to IT was interesting in itself, as one of the few occurrences of tag Y2 revealed in the concordance:

I’m a bit anxious about trying any new gadget so I suppose my old computer would probably be the answer, as long as it doesn’t interfere with any of the other programs.

This was a useful contribution to one of the questions posed in the interview, which was whether, for the scheduled evaluation of the SAMS software, the respondents would accept a new laptop with the SAMS software pre-installed. We hoped for a ‘yes’ but it turned out that, like respondent 10,

many respondents found this unacceptable; they wanted to continue to use their own computers.

We also explored the privacy/security concern using the same technique. In this case, a different six USAS semantic tags were used, those that best corresponded to privacy and security:

- | | |
|-------|----------------|
| A10- | Closed/Hiding |
| A15- | Danger |
| E6- | Worry/Concern |
| G2.1- | Crime |
| G2.2- | General ethics |
| S7.4+ | Permission |

As above, the tags were used to profile the respondents (Figure 4). The y-axis again represents the relative frequency of occurrence of the privacy- and security-related tags in the respondents' interview transcripts. The overall mean for the cumulative relative frequency of the six tags was 0.51. Thus, these tags were relatively over-represented in the responses of respondents 3, 5, 9, 10, 11 and 13.

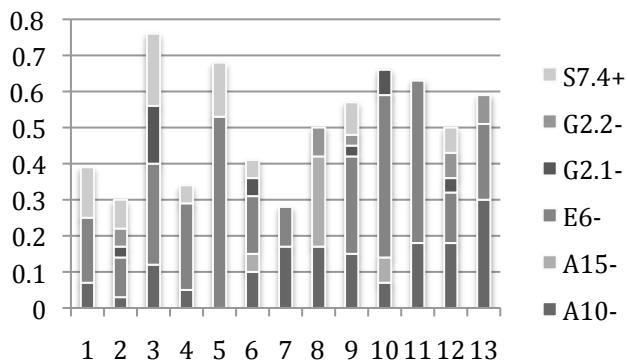


Fig 4 Respondents' privacy/security concern profiles

Each tag yielded a different set of terms in the transcripts. For example, E6 included *anxious*, *trust* and *reassuring*, while G2.2 yielded *misuse*, *sinister* and *immoral*. The relevance of the terms was easily verified by inspecting the concordances. For example, the following passage confirms the relevance of two instances of terms yielded by E6-:

[illegal access] might have consequences as far as a financial sense I'm concerned or something like that, which I'm inclined to be worried about.

The same tag (E6-) dominates the responses of 5 and 11 but their concordances reveal only general worry about dementia rather than about SAMS' implications for security and privacy. Respondents 3, 9, 10 and 13 thus appeared relatively concerned about privacy and security and this was broadly consistent with the findings of the manual analysis discussed earlier.

For the post-interview transcripts, problem domain issues (*medicines and medical treatment* and *disease*) predominate over solution domain (*information technology and computing*) as shown in Figure 2. The category *the media: newspapers, etc.* also occurs frequently, reflecting news stories providing cues for contributions.

2) Hypothesis-driven Analysis

The hypothesis-driven analysis sought to drill into the transcripts to find information related to what we believed would determine SAMS acceptance or rejection: the respondents' feelings and attitudes towards the system, as represented by their Values, Motivations and Emotions (VME) [33]. As before, manual and supervised semi-automatic analysis were performed, but this time the manual analysis was performed using eMargin [17] and the semi-automatic analysis was performed primarily to benchmark its performance.

eMargin is a collaborative annotation tool that allows the analyst to colour-mark passages of text, attach tags and share the marked-up document with other users.

The manual VME analysis investigated respondents' feelings and attitudes towards the system as well as providing evidence from emotions about possible acceptance or rejection of the system and the likelihood that system advice would be followed.

In the interviews the most commonly expressed emotion was anxiety (46.6 %), followed by distress and frustration (both 20%). Altruism, to take part in the research, was the most common motivation (52%), while privacy/security was the most frequently expressed value (72%). In the post-interview audio recordings, distress (35.2% of all emotions) and sadness (29.4%) emerged as the most frequent categories. However, motivations and values were rarely mentioned (4 and 7 total utterances), probably because the post-interview instructions biased the respondents towards reflection on their own experience and news stories.

Taking a high frequency of emotional expression to signify concern and hence an increased likelihood to adopt SAMS, all respondents apart from 4, 6, 7, 12 and 13 may be positively motivated towards the system. Respondent 13 had a poor reaction to the scenarios and low emotional engagement. Infrequent expression of emotion could indicate poor commitment to SAMS since these individuals are not motivated; whereas people who frequently express distress and sadness are probably better motivated to check their own health.

TABLE III
RESPONDENTS' FREQUENCY OF EMOTIONAL UTTERANCES

| Respondent no. | Interview emotion | Post interview emotion | Net affect |
|----------------|-------------------|------------------------|------------|
| 1 | 0 | 6 | 6 |
| 2 | 5 | 1 | 6 |
| 3 | 5 | 3 | 8 |
| 4 | 0 | No data | 0 |
| 5 | 4 | 8 | 12 |
| 6 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 |
| 8 | 3 | 4 | 7 |
| 9 | 4 | 4 | 8 |
| 10 | 4 | No data | 4 |
| 11 | 3 | 7 | 10 |
| 12 | 1 | 1 | 2 |
| 13 | 1 | No data | 1 |
| Totals | 30 | 34 | 64 |

In the manual analysis, eMargin was used to annotate the text where respondents expressed values, motivations or emotions. The use of eMargin made the manual analysis available as a reference – a gold standard – against which the results of the Wmatrix analysis could be compared.

A subset of the USAS semantic tags were isolated that corresponded to VME. None were found to be satisfactory analogues for value or motivation, but the following were identified for emotion:

| | |
|----|--------------------|
| E3 | Calm/Violent/Angry |
| E4 | Happy/Sad |
| E5 | Fear/Bravery/Shock |
| E6 | Worry/Concern |

In total, there were 275 instances of terms tagged E3, E4, E5 or E6, of which 75 corresponded to text marked up by the manual analysis, representing 27% precision. Only 25 of the manually marked-up passages were missed, giving recall of 75%.

C. Summary of Requirements

Reviewing the contributions of the manual and automated analysis, it is apparent that the 'traditional' manual analysis in section A identified the major user goals and non-functional requirements. However, only a few values, motivations and emotions were found even though the analyst was expert in such analysis and actively sought these insights. The more systematic manual discourse function analysis produced new findings on individual differences in user reactions to both the key NFR privacy and to design options in scenarios. A sub-group of users emerged (3, 4, 5, 9 & 13) who we considered likely to be unwilling adopters of SAMS. The manual VME analysis discovered another sub-group of users who showed little emotional reaction, which we considered unusual given the very real prospect of dementia affecting the lives of our senior citizen interviewees. We therefore interpreted users (4, 6, 7, 12 & 13) as another group of unwilling adopters.

The text mining, described in section B found another user sub-group of low responders (2, 4, 8 & 10) who, apart from respondent 4, did not intersect with the other two groups of potential unwilling adopters. We conjecture that low frequencies of dominant semantic categories, when these are oriented towards the solution domain, might indicate a more benevolent attitude to the system. These individuals also asked fewer questions in the interviews. Text mining of semantic categories corresponding to the privacy/security concern, supported the manual analysis identification of the privacy-sensitive user sub-group. Manual analysis of emotion was also confirmed by text mining the corresponding categories with a recall of 75%. These results give us confidence that automated text-mining tools can replicate a manual analysis even in difficult areas of sentiment. The automated analysis also produced tags that posed further analysis questions when tags were followed to utterances in the source text, as illustrated in the Computing Technology concerns.

The systematic manual analysis, excluding transcription time, took approximately 45 hours for coding. The informal audio-only analysis took 18 hours listening to interview recordings while making notes. In contrast, automated analysis was rapid, accounting for 6 hours including data cleaning and producing the results. The text-mining tools therefore afforded a considerable time saving while producing results of similar

quality. As data volumes scale up this advantage will become more significant.

The requirements implications of the user sub-groups lies in customisation. For the privacy-resistant sub-group we will restrict monitoring to quantitative measures in e-mail text, so no semantic interpretation is undertaken. This, in combination with a user control to temporarily disable monitoring, will be explained to reassure this sub-group. The low emotion sub-group will need increased motivation. Here we will employ video explanation and persuasive technology design guidelines [11], e.g. use of praise, personal address, etc. to increase user commitment to SAMS.

VI. DISCUSSION

A. Threats to Validity

Internal threats to the validity of the analysis were minimised by having the first author running the workshops and interviews and the subsequent manual analysis, while the other three authors variously applied the Wmatrix tool. This meant that the transcripts were processed blind by the tool users, avoiding bias regarding what to look for and where.

Some of the themes that emerged during the workshops, interviews and post-interview sessions conflated requirements issues for research and for what we hope will eventually be a deployed system. For example, altruism was a motivation of the respondents who volunteered to provide requirements information for SAMS, but their altruism was directed towards the research project and was not indicative of whether an altruistic person would be more or less likely to use the SAMS software. This confusion was hard to eliminate and adds noise to the already difficult problem of interpreting the data on affect.

B. SAMS Requirements

The apparent correlation between questioning and negative valence may offer an insight into who may and may not be potential users of SAMS. Similarly, we have hypothesised that the people who articulated more emotion are more likely to adopt SAMS. The affect analysis elicited user sub-groups that have implications for customisation; reassure one group worried about privacy and increase the motivation of the other low emotion group. Text mining for user profiles may have further potential in personal RE [31], for instance in eliciting individual user goals. Clearly there were limitations on the applicability of text mining. For example, apart from privacy, automated identification of motivations and values was not possible since few explicit lexical markers of these categories appear in text. Despite these caveats, the exercise did reveal important requirements about privacy and alert types that may be characteristic of senior users.

C. Methodology

The phased elicitation sessions served the important purpose of allowing us to focus on issues that emerged in earlier sessions. The interviews helped explore issues that emerged in the workshops. The self-recorded sessions, were intended to encourage respondents to provide thoughtful

insights outside the structure of a formal elicitation event, and they were somewhat successful in providing information about affect.

The analysis to which we subjected the recorded text provided some interesting data with one unanticipated correlation between questioning and negative valence. This was time-consuming work, even for the relatively modest number of respondents, and it is possible that further tool support, such as sentiment analysis to generate the valence results, may be of use here in future.

The observed performance of Wmatrix needs to be treated carefully. At first glance, frequency profiling appeared to give very different results from the manual discourse analysis. This was because of the differences between the tailored set of categories used in the discourse analysis and the general-purpose nature of the USAS tags. The fact that Wmatrix is applied to individual terms or multi-word units, while the discourse analysis is classified at the paragraph level, also contributes to the difference. However, by providing an alternative way of viewing the text, purely as a data set, frequency profiling was able to complement the manual analysis, finding different ways to contrast the respondents' contributions, allowing the analyst to follow the evidence or even serendipitously following their own hunches into the text by clicking on tag terms to view a set of concordances.

Filtering on tag subsets that map on to particular classes of information is limited by the general-purpose nature of the tagset. No tags mapped on to value or motivation, for example. Where analogues were found, however, performance was reasonable and suggests that there is potential for similar tools.

With respect specifically to identification of affect-laden utterances, if the observed level of performance (27% precision, 75% recall) proved to be replicated in future analyses of affect, it would be tolerable provided the following caveats held:

- The assessment was done across a sufficiently wide and representative user population; and
- Developing an understanding of the *range* of emotional responses was more important than collection of *all* responses.

Thus, rapidity of analysis would compensate for having only qualified confidence in the result. Such a trade-off is inherent for applications of text mining but it means that text mining is not suitable in all RE contexts. In particular, it should be noted that while easy access to terms' context in the text using concordances makes validation of true positives and rejection of false positives easy, only a painstaking manual analysis can reveal the false negatives. If this needs to be done, it negates any effort-saving advantages of text mining [4]. Nevertheless, text mining did allow us to posit heuristics which may direct future automated analysis of affect-laden applications, e.g. screen users for negative emotional reactions, low frequency of emotions, and sensitivity to key values.

For comparison, with the privacy and security tags (Figure 4), 63 of the 172 instances of terms returned by the tags proved to be of relevance, giving a precision figure of 37% when we checked each instance. Since eMargin mark-up had not been

used for the data-driven analysis, we were unable to quantify recall.

The implication of this is that, of the responses plotted for each respondent in Figure 4, only a third are likely to be a true positive, i.e. relevant to understanding the privacy/security concern. However, since each column is made up of the same tags, and if we assume that the precision was consistent across each tag for each respondent, the conclusions we drew from the data in the graph remain robust.

As usual for text mining techniques, there is an inverse relationship between recall and precision; recall can be increased by including more tags, but at the expense of precision. In RE applications, recall is generally favoured over precision, since errors of omission are harder to detect than errors of commission. Seen in this light, the performance of Wmatrix was adequate, permitting hypotheses to be checked quickly.

A final reflection concerns the scenario used for text mining directed RE. As applications come to be delivered more frequently over the web, and as requirements are also analysed by remote web capture, the collection of user feedback to mock-ups and prototypes may become commonplace. Given high numbers of potential users and corresponding diversity of needs, automated analysis of linguistic feedback, opinions and affect will begin to pay off.

VII. CONCLUSIONS

We set out to develop a tailored requirements discovery method that addressed critical questions of technology acceptance for affect-laden problems. We also wanted to exploit tool support to help our analysis. In the end we gained insights into the myriad ways in which affect colours peoples' views of health-related problems and systems, and we gained insights into how seniors regard technology as risky, and their concerns for privacy and security. While the implications for customisation we have drawn from this analysis will need to be validated by user reaction to SAMS in practice, the process we have undertaken has potential for application in other domains where motivation and affect are important.

We have accumulated information that has allowed us to better understand the requirements for SAMS. We have also gained insights into the potential and limitations of shallow semantic analysis for RE.

In the next phase of the project, we will be developing the SAMS software and subjecting it to trials that are intended to reveal more about the extent to which it will prove acceptable before deploying SAMS in a longitudinal study of real users.

ACKNOWLEDGMENTS

The work described in this paper is funded by EPSRC project ref. EP/K015796/1 *Software Architecture for Mental Health Self Management (SAMS)*.

REFERENCES

- [1] Alwan, M., Mack, D.C. et al. (2006), "Impact of passive in-home health status monitoring technology in home health: Outcome pilot" in *Proceedings 1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare, D2H2-06*. pp.79-82.

- [2] Ballard C. and Corbett A. (2012), “Screening for dementia: An opportunity for debate”, *Expert Rev Neurother.* vol. 11(10), pp. 1347-9.
- [3] Beckles, B. (2005), “User requirements for UK e-Science grid environments” in *Proceedings 4th UK All Hands E-Science Meeting*.
- [4] Berry, D., Gacitua, R., Sawyer, P., Tjong, S.F. (2012), “The case for dumb requirements tools” in *Proceedings 18th International Working Conference on Requirements Engineering: Foundations of Software Quality (REFSQ'12)*, pp 211-217.
- [5] Blei, D.M., Ng, A.Y. and Jordan, M.I. (2003), “Latent dirichlet allocation”, *Journal of Machine Learning Research*, vol. 3, pp 993-1022.
- [6] Callele, D., Neufeld E. and Schneider, K. (2009), “Augmenting emotional requirements with emotion markers and emotion prototypes”, in *Proceedings, RE 2009*. Los Alamitos CA: IEEE Computer Society Press, pp. 373-374.
- [7] Cleland-Huang, J., Berenbach, B., Clark, S., Settimi, R., Romanova, E. (2007), “Best practices for automated traceability”, *IEEE Computer*, vol. 40, pp 27-35.
- [8] Cysneiros, L.M., (2002), “Requirements engineering in the health care domain”, *Proceedings 10th IEEE Int. Conf. on Requirements Engineering*. Los Alamitos CA: IEEE Computer Society, pp. 350-356.
- [9] Davis, F.D. (1989), “Perceived usefulness, perceived ease of use, and user acceptance of information technology”, *MIS Quarterly*, vol. 13(3), pp. 319-340.
- [10] Dumais, S., Furnas, G., Landauer, T., Deerwester, S. and Deerwester, S. (1995), “Latent semantic indexing”, *Proceedings. 4th Text REtrieval Conference (TREC-4)*.
- [11] Fogg, B.J. (2003), *Persuasive Technology: Using Computers to Change What We Think and Do*, San Francisco: Morgan Kaufmann.
- [12] Gacitua, R., Sawyer, P. and Gervasi, V. (2011), “Relevance-based abstraction identification: Technique and evaluation”, *Requirements Engineering*, vol. 16(3), pp. 251-265.
- [13] Garde, S. and Knaup, P. (2006), “Requirements engineering in health care: The example of chemotherapy planning in paediatric oncology”, *Requirements Engineering*, vol. 11, pp 265-278.
- [14] Hayes, J.H., Dekhtyar, A., Sundaram, S.K. (2006), “Advancing candidate link generation for requirements tracing: The study of methods”, *IEEE Transactions on Software Engineering*, vol.32, pp.4-19.
- [15] Jiroka, M., Procter, R. et al. (2005), “Collaboration and trust in healthcare innovation: The eDiaMoND case study”, *Journal of Computer-Supported Cooperative Work*, vol. 14(4), pp. 369-398.
- [16] Jorgensen, J.B. and Bossen, C. (2003), “Requirements Engineering for a pervasive health care system” in *Proceedings 11th IEEE International Conference on Requirements Engineering (RE'03)*. pp. 56 – 64.
- [17] Kehoe, A. and Gee, M. (2012), “eMargin: A collaborative text annotation tool.” in *Proceedings 33rd Conference on International Computer Archive of Modern and Medieval English (ICAME 33)*.
- [18] Le, X., Lancashire, I., Hirst, G. and Jokel, R. (2011), “Longitudinal detection of dementia through lexical and syntactic changes in writing: A case study of three British novelists”, *Literary and Linguistic Computing*, vol. 26(4), pp. 435-461.
- [19] McArthur, T. (1981), *Longman Lexicon of Contemporary English*. London: Longman.
- [20] Rodriguez-Ferreiro, J., Davies, R., González-Nosti, M., Barbón, A. and Cuentos, F. (2008), “Name agreement, frequency and age of acquisition, but not grammatical class, affect object and action naming in Spanish speaking participants with Alzheimer’s disease”, *Journal of Neurolinguistics*, vol. 22(1), pp 37-54.
- [21] Nasreddine, Z. et al. (2005), “The Montreal Cognitive Assessment, MoCA: A brief screening tool for mild cognitive impairment”. *Journal of the American Geriatrics Society*, vol. 53(4), pp. 1532-5415.
- [22] Perrone, V., Finkelstein, A. et al.. (2006), “Developing an integrative platform for cancer research: A requirements engineering perspective”, *Proceedings. 5th UK All Hands e-Science Meeting*.
- [23] Pilealm, S. and Timpka, T. (2008), “Third generation participatory design in health informatics: Making user participation applicable to large-scale information system projects”. *Journal of Biomedical Informatics*, vol. 41, pp. 327-339.
- [24] Popescu, D., Rugaber, S., Medvidovic, N. and Berry, D.M. (2008), “Reducing ambiguities in requirements specifications via automatically created object-oriented models” in Paech, B., Martell, C. (eds) *Innovations for requirement analysis: From stakeholders' needs to formal designs*. Heidelberg: Springer, pp. 103–124.
- [25] Ramos, I. and Berry, D.M. (2005), “Is emotion relevant to requirements engineering?”, *Requirements Engineering*, vol. 10(3), pp. 238-242.
- [26] Ramos, I., Berry, D.M. and Carvalho, J. (2002). “The role of emotion, values and beliefs in the construction of innovative work realities” in *Proceedings of SoftWare 2002: Computing in an Imperfect World*.
- [27] Rayson, P. (2008), “From key words to key semantic domains”, *International Journal of Corpus Linguistics*, vol. 13(4), pp. 519-549.
- [28] Rayson, P., Archer, D., Piao, S.L., McEnery, T. (2004), “The UCREL semantic analysis system”. In *Proceedings of the Workshop on Beyond Named Entity Recognition Semantic labelling for NLP tasks, in association with 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pp. 7-12.
- [29] Ryan, K. (1993), “The role of natural language in requirements engineering” in *Proceedings 1st Int. Symposium on Requirements Engineering*. Los Alamitos CA: IEEE Computer Society, pp. 240-242.
- [30] Sawyer, P., Rayson, P. and Cosh, K. (2005), “Shallow knowledge as an aid to deep understanding in early-phase requirements engineering”, *IEEE Transactions on Software Engineering*, vol. 31(11), pp. 969-981.
- [31] Sutcliffe, A.G., Fickas, S. and Sohlberg, M.M. (2006). “PC-RE: A method for personal and contextual requirements engineering with some experience,” *Requirements Engineering*, vol. 11, 157-163.
- [32] Sutcliffe, A.G. and Sawyer, P. (2013), “Requirements elicitation: Towards the unknown unknowns” in *Proceedings 21st IEEE International Conference on Requirements Engineering*, pp. 92-104.
- [33] Sutcliffe, A.G., Thew, S. et al. (2010), “User engagement by user-centred design in e-health”, *Philosophical Transactions of the Royal Society, special issue on e-Science, series A*, vol. 368, pp. 4209-4224.
- [34] Thew, S. and Sutcliffe, A.G. (2008). “Investigating the role of soft issues in the RE process” in *Proceedings 16th IEEE International Requirements Engineering Conference*, pp. 63-66.
- [35] UK Department of Health (2013), “G8 dementia summit communiqué” <https://www.gov.uk/government/publications/g8-dementia-summit-agreements>. Accessed 10 March 2014
- [36] Wilson, T., Wiebe, J. and Hoffmann, P. (2005), “Recognizing contextual polarity in phrase-level sentiment analysis” in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 347-354.

Hidden in Plain Sight: Automatically Identifying Security Requirements from Natural Language Artifacts

Maria Riaz, Jason King, John Slankas, and Laurie Williams

Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA

[mriaz, jtking, john.slankas, laurie_williams]@ncsu.edu

Abstract—Natural language artifacts, such as requirements specifications, often explicitly state the security requirements for software systems. However, these artifacts may also imply additional security requirements that developers may overlook but should consider to strengthen the overall security of the system. *The goal of this research is to aid requirements engineers in producing a more comprehensive and classified set of security requirements by (1) automatically identifying security-relevant sentences in natural language requirements artifacts, and (2) providing context-specific security requirements templates to help translate the security-relevant sentences into functional security requirements.* Using machine learning techniques, we have developed a tool-assisted process that takes as input a set of natural language artifacts. Our process automatically identifies security-relevant sentences in the artifacts and classifies them according to the security objectives, either explicitly stated or implied by the sentences. We classified 10,963 sentences in six different documents from healthcare domain and extracted corresponding security objectives. Our manual analysis showed that 46% of the sentences were security-relevant. Of these, 28% explicitly mention security while 72% of the sentences are functional requirements with security implications. Using our tool, we correctly predict and classify 82% of the security objectives for all the sentences (precision). We identify 79% of all security objectives implied by the sentences within the documents (recall). Based on our analysis, we develop context-specific templates that can be instantiated into a set of functional security requirements by filling in key information from security-relevant sentences.

Index Terms— Security, requirements, objectives, templates, access control, auditing, text classification, constraints, natural language parsing.

I. INTRODUCTION

Security requirements provide a foundation for building secure software systems. Despite the availability of methods and processes for security requirements engineering [1], teams often do not focus on security during early stages of software development [2]. Existing approaches outline the various steps involved in identifying security requirements, but leave the task of executing these steps for the requirements engineer, who may not be an expert in security. Many security requirements are functional in nature and need to be incorporated into system design to ensure data and system security. Natural language requirements artifacts, such as requirements documents, often

explicitly state the security requirements for software systems. However, additional sentences in these documents may have security implications [3] leading to additional security requirements.

Our research on identifying applicable security requirements for software systems is guided by the following primary motivations: Software systems that share security objectives, such as confidentiality or integrity, also have similar sets of security requirements [4]. These security requirements, if specified at the right level of abstraction, can be reusable across multiple systems, even as a set to meet the same security objective [5, 6]. Patterns and similarities in grammar or phrasing of security requirements may exist and allow the security requirements to be reused across multiple software systems with minor tweaks to content, such as different actions taken or different resources being acted upon. By first identifying both the explicit and implied security objectives of a software system, we intend to discover an abstract set of security requirements that may be considered when developing any software system that shares similar security objectives. Natural language requirements artifacts often contain security-relevant sentences that are indicative of the security objectives and security requirements of the system [3].

The goal of this research is to aid requirements engineers in producing a more comprehensive and classified set of security requirements by (1) automatically identifying security-relevant sentences in natural language requirements artifacts, and (2) providing context-specific security requirements templates to help translate the security-relevant sentences into functional security requirements.

We present a tool-assisted process, Security Discoverer (SD), that incorporates machine learning techniques to identify a set of security requirements for an input set of natural language requirements artifacts. We classify the sentences in the input in terms of their security objectives (such as, confidentiality, integrity, availability). This classification can be used to guide the analyst in creating an appropriate set of security requirements and in organizing the resultant set of security requirements. Using this approach, we have classified 10,963 sentences in six natural language artifacts from the electronic healthcare domain in terms of their corresponding

security objectives. By observing similarities and abstracting common elements in the classified set of security-relevant sentences, we empirically derive a set of context-specific security requirements templates. Our tool suggests applicable templates for instantiation by the requirements engineers to generate the security requirements.

For example, consider the sentence "The system shall provide a means to edit discharge instructions for a particular patient."¹ This sentence does not explicitly state a security requirement but implies security requirements for confidentiality (*of patient's discharge instructions*), integrity (*when editing*) and accountability (*who performed the edits*). Security requirements that can be generated by instantiating corresponding templates include:

- "The system shall enforce access privileges that enable authorized users to edit discharge instructions for a particular patient." (confidentiality)
- "The system shall log every time discharge instructions for a particular patient are edited." (accountability)

We use the following research questions to guide us in meeting our research goal:

RQ1: What are the core categories of security objectives in existing literature that should be considered during the security requirements engineering process?

RQ2: How often are security objectives explicitly stated or implied in natural language requirements artifacts?

RQ3: How effectively can security objectives be identified and extracted from natural language project documents?

RQ4: What similarities (words, phrases, grammatical structure, etc.) exist among security-relevant sentences for each security objective?

RQ5: What common templates for specifying functional security requirements can be empirically derived from the security-relevant sentences?

Our research contributes the following:

- A set of core categories of security objectives that requirements engineers should consider during the requirements engineering process.
- A tool-assisted process to aid requirements engineers in identifying and classifying security relevant-sentences in terms of security objectives.
- A set of context-specific security requirements templates to help requirements engineers translate classified security-relevant sentences into functional security requirements that meet specific objectives.

The rest of this paper is organized as follows: Section II reviews the background and related work. Section III presents the security objectives to address RQ1 followed by our proposed tool-assisted process, Security Discoverer, in Section IV. In Section V, we describe our research methodology. Section VI presents results and evaluation to address RQ2-RQ4. We present the set of context-specific templates based on our analysis in Section VII to address RQ5. Section VIII

discusses threats to validity for our study. Finally, Section IX concludes the paper in addition to outlining future directions.

II. BACKGROUND AND RELATED WORK

In this section, we discuss the background of security objectives, and related work in requirement classifications and security requirements engineering.

A. Security Objectives and Requirements

Security objectives are the security goals or desired security properties of a system [7]. Security requirements are functional and non-functional requirements that operationalize security objectives without specifying how to achieve those objectives. Functional security requirements describe the desired security behavior of a system [8] and, if incorporated, can achieve the corresponding security objectives. For this paper, we use the term security requirements to mean functional security requirements.

Firesmith [9] argues security requirements can be reusable across multiple systems and has proposed the use of parameterized templates to model reusable security requirements. Mellado et al., [6] argue the effectiveness of reusing related security requirements that act as a group to meet security objectives. In our work, we group security-relevant sentences in terms of security objectives and provide context-specific templates to meet these objectives, building on the observations from previous work.

B. Identifying Security Requirements

Security Requirements Engineering (SRE) has gained focus in recent years with emphasis on identifying security requirements early on in the software development lifecycle. Mellado et al. have conducted a systematic review of SRE approaches [10] to summarize existing methodologies. Fabian et al. also provide a comparison of SRE methods [1]. These methods include lifecycle-based approaches such as Microsoft SDL [11] as well as methods that solely focus on SRE, such as the SQUARE method [12], which provides a framework for generating non-functional security requirements. Other approaches for identifying security requirements include misuse or abuse cases [13], anti-goals [14], and assurance arguments [15].

A recent analysis of various SRE methods [16] indicates that despite the availability of a number of SRE methods, only a handful of these methods have been used in practice. Their findings, based on feedback from practitioners across various organizations, also highlight that SRE as a tool-assisted process may facilitate secure software development. Efforts to automate parts of SRE process have resulted in organizational learning approach to SRE [17] that identifies when a security requirement is added to a natural language artifact. This helps build a repository of security requirements for consideration and reuse in subsequent projects. Our process goes beyond identifying explicit security requirements in text. We identify implied security-relevant sentences in natural language artifacts and associated security objectives. We also provide context-specific templates to identify functional security requirements to meet the identified objectives.

¹ <http://www.hl7.org/>

Another focus area related to security requirements engineering has been on extracting security requirements from regulatory texts ([18, 19]). Other researchers have explored using natural language to generate access control policies ([20, 21]). The focus of our research is on extracting security requirements from existing functional requirements and requirements-like documents and we do not consider issues related to regulatory compliance or policy specification.

C. Requirements Classification

While text classification, especially with regard to Term Frequency - Inverse Document Frequency (TF-IDF), has been studied for a relatively long period of time [22], non-functional requirement (NFR) classification first appeared in the literature in 2006 [23]. In their work, Cleland-Huang et al. applied TF-IDF with an additional parameter to specify the frequency of indicator terms for a NFR category as compared to the appearance of those terms in the requirement currently under test. Their work performed well with a 0.8129 recall, successfully identifying 81% of the NFRs in the dataset. However, their precision was 0.1244 indicating a large number of false positives. Other researchers [24, 25] built upon this work by using the same dataset as Cleland-Huang, but adopt naïve Bayes and Support Vector Machine (SVM) classifiers. Both experiments reported higher scores for precision than the original research. We evaluated a number of different machine learning algorithms for text classification and decided upon a k -NN classifier as its performance matched that of an SVM classifier and analysts can more easily understand where results were derived in the k -NN classifier.

III. SECURITY OBJECTIVES OF SOFTWARE SYSTEMS

By identifying the security objectives expressed or implied by a particular sentence within a document, we gain an understanding of the intent of the sentence as well as possible requirements and mechanisms to establish that intent. Security objectives of software systems involve not only technical aspects from system development perspective but also operational and management aspects. For the purpose of this research however, we focus on technical security objectives of software systems. While certain sets of security objectives are widely known such as "Confidentiality, Integrity, and Availability (CIA) Triad", we want to ensure the completeness of our security objective set.

RQ1: What are the core categories of security objectives in existing literature that should be considered during the security requirements engineering process?

We examined six security standards ([8, 26-30]), two taxonomies of security objectives and requirements ([9, 14]) and two security seminal papers and books ([7, 31]). We created a list of security objectives identified from the above sources. We define each of the technical security objectives below. The references from where these objectives have been identified are listed after the objective's name. We also provide example sentences from the set of documents we analyzed (see Section V.A) that indicate the presence of corresponding

objective. The examples are numbered as: <Document ID>-<Security Objective Abbreviation>.<#>.

Confidentiality (C) ([7-9, 14, 26, 29]): The degree to which the "data is disclosed only as intended" [7]

Example: "The system should provide the ability to electronically capture patient data including medications, vital signs, and other data as structured data" (EA-C.1)

Integrity (I) ([7-9, 14, 26, 29]): "The degree to which a system or component guards against improper modification or destruction of computer programs or data." [28]

Example: "...the system shall provide the ability to mark the information as erroneous in the record of the patient in which it was mistakenly associated and represent that information as erroneous in all outputs containing that information." (ED-I.1)

Identification & Authentication (IA) ([8, 9, 14, 26, 29]): The need to establish that "a claimed identity is valid" for a user, process or device. [27]

Example: "The system shall authenticate the user before any access to Protected Resources (e.g. PHI) is allowed, including when not connected to a network e.g. mobile devices." (CT-IA.2)

Availability (A) ([7-9, 14, 26, 29]): "The degree to which a system or component is operational and accessible when required for use." [32]

Example: "Provides business continuity in the situation where the EHR system is not available by providing access to the last available clinically relevant patient data in the EHR." (NU-A.3)

Accountability (AY) ([7-9, 14, 26, 29]): Degree to which actions affecting software assets "can be traced to the actor responsible for the action" [7]

Example: "Every entry in the health record must be identified with the author and should not be made or signed by someone other than the author." (ED-AY.1)

Privacy (PR) ([8, 9, 14]): The degree to which an actor can understand and control how their information is used.

Example: "Nurses need to provide legitimate care in crisis situations that may go against prior patient consent directives ("break the glass" situations)". (NU-PR.2)

IV. SECURITY DISCOVERER

We now present our process, Security Discoverer (SD), and its associated tool.

A. Overview

We have developed a four-step tool-assisted process for identifying security requirements. As a first step, our tool takes natural language requirements artifacts (requirement specifications, feature requests, etc.) and a trained classifier for the current problem domain as input. The tool parses the artifacts as text sentences and identifies which (if any) security objectives relate to each sentence. The tool then presents the user with a list of applicable security requirements templates for the identified objectives. The user then selects the

appropriate templates and completes the template with details from the initial requirement. Our tool² provides an authoring mechanism to finalize the identified security requirements. The tool also supports traceability of generated security requirements to source sentences in input artifacts. We explain the steps of the SD process below.

B. Step 1: Pre-process Artifacts

We import a natural language requirement artifact into the SD tool to prepare each sentence to be classified with relevant security objective(s). A user first needs to convert the artifact into a text only format. Generally, this conversion can be accomplished through the “Save As” format within Microsoft Word or other document applications. As such, this process will not convert tables or images properly and the user will need to manually perform the conversion for those sections. Once the artifact has been prepared, the tool will first read the entire text into the system. Next, to provide additional context and features for the classifier, the tool applies a concise document grammar (Figure 1) to label each sentence in the text to a specific type:

- *title*: Sentences that follow capitalization rules for titles.
- *list start*: These sentences represent the header or description of a list that follows.
- *list element*: These sentences represent individual items contained within an ordered or unordered list. These sentences are combined with the start of the list when sent to the parser and for classification. Combining the two provides additional context to both human analysts and machine classifiers.
- *normal sentence*: These sentences are not considered as titles, list starts, or list elements.

Further, we identify heading and list identifiers (e.g., “4.1.1” and “•”) and remove those identifiers from sentences used in classification. These identifiers create superficial differences among sentences and can possibly skew classification results. When sentences are classified in the next step, we combine “list start” with each identified “list element” as part of the same list to provide additional context.

Within Figure 1, italicized words represent nonterminal symbols that can be replaced by other symbols on the right-hand side. Words in normal font are terminal symbols. Characters within quotation marks are also specific terminal symbols. λ represents an empty expansion of a nonterminal.

| | | |
|--------------------|---|---|
| <i>Document</i> | → | <i>Line</i> |
| <i>Line</i> | → | <i>listID title line</i> <i>title line</i> <i>sentence line</i> λ |
| <i>sentence</i> | → | <i>normalSentence</i> <i>listStart</i> (“.” “-”) <i>listElement</i> |
| <i>listElement</i> | → | <i>listID sentence listElement</i> λ |
| <i>listID</i> | → | <i>listParanID</i> <i>listDotID</i> <i>number</i> |
| <i>listParanID</i> | → | “(” <i>id</i> “)” <i>listParanID</i> <i>id</i> “)” <i>listParanID</i> λ |
| <i>listDotID</i> | → | <i>id</i> “.” <i>listDotID</i> λ |
| <i>id</i> | → | <i>letter</i> <i>romanNumeral</i> <i>number</i> |

Fig. 1. Document Grammar

While we do not expect the documents to be well-formed, our process works better with shorter, well-formed sentences.

² Source code available at: <http://go.ncsu.edu/securitydiscoverer/>

C. Step 2: Classify for Security Objectives

We classify each sentence in the input into zero or more security objectives. The classifier can be created in one of three ways: 1) training a new classifier by manually classifying sentences for security objectives from related projects; 2) utilizing an existing classifier; or 3) utilizing the tool in an interactive fashion to provide recommendations for classifications to aid the manual process.

We utilize a *k*-NN classifier for this step. Such classifiers work by taking a majority vote of the existing classifications of the *k* nearest neighbors to the item under test. To determine the closest sentence(s), we apply a custom distance function based upon a modified version of Levenshtein distance [33]. Rather than using the resulting number of edits to transform one string into another as the value as the Levenshtein distance does, our metric computes the number of word transformations to change one sentence into another. Repetition of words and phrases in different sentences can help the classification process. While other machine learning algorithms can provide similar performance to *k*-NN classifier, the *k*-NN classifier provides for easier interpretation of results by requirements engineers as they can see similar sentences and associated classifications. The distance metric can also be used within distance-based clustering algorithms for further analysis.

Once the classification is complete, the user may review the predicted security objectives for the security-relevant sentences. If necessary, the user can correct the classified objectives within the tool.

D. Step 3: Select Context-specific Templates

Once the security objectives have been identified for a given sentence, the tool presents the user with a list of context-specific security requirements templates for the security objectives and values (such as action or time) present within the sentence. The security templates are further discussed in Section VII. A sample template for the objective “accountability” is displayed in Table VI. The user selects which templates apply to the given sentence. SD tracks which templates have been selected. The usage data provides the ability to determine which templates are most frequently used and in what combination. Additionally, the data could be used within a recommendation engine in future versions of the tool.

E. Step 4: Generate Requirement Sentences

Once the requirement templates have been selected by the user, the tool presents the requirements text in an editor text window for the user to complete. In situations where a replaceable value has been found, the replacement is already made. For instance, if an availability-related sentence specifies “during business hours”, tool detects the time period and would automatically place that phrase into the generated requirement template. The tool maps generated requirements to source sentences to produce a traceability matrix.

V. RESEARCH METHODOLOGY

In this section, we discuss our methodology for collecting and preparing the selected documents for use within our study.

A. Study Documents

We have selected six freely-available natural language requirements artifacts from the electronic healthcare domain, listed in Table I. Due to regulation and standardization, documents in healthcare domain generally tend to be well-formed. However, we select a variety of document types, including feature requests that are not well-formed. The selected documents are from USA and Canada. Use of different spellings for same words (e.g., color vs. colour) may also affect the performance of our classifier.

B. Study Oracle

We have developed a study oracle to train our tool and evaluate the performance of our classifier. To create the study oracle, three researchers manually read each natural language sentence in the six healthcare documents, and classify the sentence with relevant security objectives as follows:

- 1) Convert the document into text-only format.
- 2) Import one text document into SD Tool, and parse the document into individual sentences using natural language processing (see Section IV.B).
- 3) Manually classify each sentence in a document.
 - a) *Classification Phase*: For each document, two researchers individually classify each sentence to identify security objectives that apply to the sentence. The classification phase results in the creation of two separate output files (one per researcher) for each input document.
 - b) *Validation Phase*: A third researcher generates a difference report from the classifications of the other two researchers. This third researcher resolves the differences by communicating with the original two researchers to generate consensus for creating a final, consolidated classified corpus document.

Table I provides a document-wise breakdown of sentences classified per security objective. Each sentence could be classified in terms of zero or more security objectives. The researchers spent a total of approximately 160 person-hours to create and validate the oracle that we use for further analysis. Researchers had a moderate agreement [34] on whether a sentence was security-relevant or not (indicated by a kappa score of 0.54). Of the security-relevant sentences, we had an almost perfect agreement in terms of whether a sentence explicitly talks about security or implies a need for security (kappa score of 0.85). We also had a fair agreement on classification of objectives for each sentence (kappa score of 0.32; kappa score tends to decrease as classification categories increase). Requirements engineers looking to adopt our process can incrementally build upon our existing classifier or train a shared classifier for their domain over time by classifying security-relevant sentences in natural language artifacts. Performance of the classifier is expected to improve as the number of classified sentences increases. Incrementally

evolving the classifier as a community will save time and effort upfront while creating a knowledgebase of security-relevant sentences and security objectives of software systems.

C. Study Procedure

Once the study oracle has been created, we execute a variety of classifiers (our k -NN classifier and from Weka [35] - a multinomial naïve Bayes classifier, and a SMO - sequential minimal optimization classifier) on the document set. For each classifier considered, we tested using a stratified n -fold cross-validation and computed the precision, recall, and F_1 measure. To compute these values, we first need to categorize the classifier's predictions into three categories. True positives (TP) are correct predictions. False positives (FP) are predictions in which the sentence of another classification is incorrectly classified as the one under evaluation. False negatives (FN) are predictions in which a sentence of the same classification under evaluation is incorrectly placed into another classification. Precision (P) is the proportion of correctly predicted classifications against all predictions for the classification under test: $P = TP/(TP + FP)$. Recall is the proportion of classifications found for the current classification under test: $R = TP/(TP + FN)$. F_1 measure is the harmonic mean of precision and recall, giving equal weight to both: $F_1 = 2 \times \frac{P \times R}{P + R}$.

With the n -fold cross-validation, data is randomly partitioned into n folds based upon each fold of approximately equal size and equal response classification. For each fold, the classifiers are trained on the remaining folds and then the contents of the fold are used to test the classifier. The n results are then averaged to produce a single result. We follow Han et al.'s recommendation [36] and use 10 folds as this produces relatively low bias and variance. The cross-validation ensures that all sentences are used for training and that each sentence is tested just once. We directly utilized Weka classifiers through the available Java APIs utilizing their default options. Since the Weka classifiers do not natively support multiple classifications for an item, we created individual classifiers for each algorithm and classification. As the folds are randomly generated, we executed the tests 3 times and averaged the results. To extract the top 20 keywords for each security objective, we utilized the information gain [37] attribute selector within Weka. Yang and Pedersen [38] found information gain to be the most effective method for feature selection in text classification.

D. Security Requirements Template Extraction

We analyze the classified set of sentences associated with each security objective and identify commonalities in those sentences based on the following attributes:

- Common patterns and themes in sentence structure
- Keywords in the sentences
- Clustering of sentences (k-medoids/LDA)

Based on our analysis, we develop templates that would allow incorporating security requirements to meet corresponding security objectives while maintaining neutrality to the mechanisms. We discuss the templates in Section VII.

TABLE I. DOCUMENTS AND ASSOCIATED SECURITY OBJECTIVE COUNTS

| Doc. ID | Document Title | #Sentences | Security Objectives | | | | | | |
|---------------------|--|--------------|---------------------|-------------------|------------------|------------------|-------------------|-----------------|-------------------|
| | | | C | I | IA | A | AY | PR | None |
| CT | Certification Commission for Healthcare Information Technology (CCHIT) Certified 2011 Ambulatory EHR Criteria ³ | 331 | 252 | 214 | 19 | 14 | 260 | 5 | 6 |
| ED | Emergency Department Information Systems Functional Document ⁴ | 2328 | 1162 | 1173 | 75 | 35 | 1354 | 76 | 773 |
| NU | Pan-Canadian Nursing EHR Business and Functional Elements Supporting Clinical Practice ⁵ | 264 | 67 | 77 | 4 | 26 | 43 | 10 | 96 |
| OR | Open Source Clinical Application Resource (OSCAR) Feature Requests ⁶ | 5081 | 696 | 974 | 104 | 10 | 1184 | 18 | 3735 |
| PS | Canada Health Infoway Electronic Health Record (EHR) Privacy and Security Requirements ⁷ | 1623 | 146 | 120 | 43 | 31 | 149 | 85 | 928 |
| VL | Virtual Lifetime Electronic Record User Stories ⁸ | 1336 | 693 | 731 | 13 | 19 | 797 | 10 | 375 |
| Total # (%): | | 10963 | 3016 (27%) | 3289 (30%) | 258 (~2%) | 135 (~1%) | 3787 (34%) | 204 (2%) | 5913 (54%) |

VI. EVALUATION

In this section, we address research questions RQ2 - RQ4.

RQ2: How often are security objectives explicitly stated or implied in natural language requirements artifacts?

Based on the study oracle, we identified that 46% of the sentences in input artifacts relate to security. Given that we selected documents from industry standards and best practices related to the healthcare domain (which involves protected health information), security-relevant sentences intuitively form a large proportion of the document. Of all the security-relevant sentences, only 28% *explicitly* mention security (13% of total sentences, similar to our earlier findings [3]), while 72% are functional requirements with security implication (an additional 33% of total sentences). If implied security objectives are not considered, requirements engineers may overlook key security requirements. Table II provides a document-wise breakdown of sentences and whether security objectives were implied or explicitly stated.

From the security relevant sentences, we identified the security objectives that are implied by each sentence. The top three implied security objectives are accountability (34% of all sentences), integrity (30%) and confidentiality (27%). Privacy (2%), identification & authentication (~2%), and availability (~1%) objectives were implied by only a small percentage of all sentences. Our results indicate that 93% of the security-relevant sentences implied more than one security objective. Table III presents the 10 most frequently occurring security objective groups. Confidentiality and accountability each appear in 7 of 10 top objective groups, suggesting that confidentiality and accountability are common security objectives for healthcare systems. Integrity appears in 6 of 10 top objective groupings.

³ <https://www.cchit.org/>

⁴ <http://www.hl7.org/>

⁵ <https://www.infoway-inforoute.ca/>

⁶ <http://oscarcanada.org/>

⁷ <https://www.infoway-inforoute.ca/>

⁸ <http://www.va.gov/vler/>

The confidentiality, integrity, and accountability objectives appear *together* in the classifications of 2,232 sentences (20% of all sentences classified), suggesting a strong relationship among the three. For example, the sentence “The system shall provide a means to edit discharge instructions for a particular patient” [ED] implies that the confidentiality of discharge instructions should be maintained since it is protected health information; that the integrity of the discharge instruction data upon editing should be maintained; and that accountability should ensure that the user editing the discharge instructions can be held responsible.

TABLE II. IMPLICIT AND EXPLICIT SECURITY-RELEVANT SENTENCES

| Doc ID | Total Sente-nces | Explicit Security # (%) | Implicit Security # (%) | Total Security # (%) | Not Security Related |
|---------------|------------------|-------------------------|-------------------------|----------------------|----------------------|
| CT | 331 | 89 (27%) | 236 (71%) | 325 (98%) | 6 (2%) |
| ED | 2328 | 274 (12%) | 1281 (55%) | 1555 (67%) | 773 (33%) |
| NU | 264 | 41 (16%) | 127 (48%) | 264 (64%) | 96 (36%) |
| OR | 5081 | 174 (3%) | 1172 (23%) | 1346 (26%) | 3735 (74%) |
| PS | 1623 | 628 (39%) | 67 (4%) | 695 (43%) | 928 (57%) |
| VL | 1336 | 185 (14%) | 776 (58%) | 961 (72%) | 375 (28%) |
| Tot-al | 10963 | 1391 (13%) | 3659 (33%) | 5050 (46%) | 5913 (54%) |

Confidentiality and accountability appear together in the classifications of 2,859 sentences (26% of all sentences classified). The act of controlling access to sensitive data to help promote confidentiality is closely tied to the act of ensuring that a complete list of users who have accessed the sensitive data may be maintained for accountability. Therefore, in our study oracle, sentences that involve create/read/update/delete actions upon sensitive data are often classified as implying both confidentiality and accountability.

Integrity and accountability appear together for 3,119 sentences (28.5% of all sentences classified). With respect to accountability, integrity helps ensure that the traces of user

activity in the system may not be corrupted, modified, or damaged so that users can always be held accountable.

Privacy and identification/authentication objectives also appear in the top ten objective groupings, but are much less common. Privacy and identification/authentication often appear in combination with confidentiality, integrity, and/or accountability objectives.

TABLE III. FREQUENTLY OCCURRING OBJECTIVE GROUPS

| Frequency # (% sec- relevant) | Objective Group |
|-------------------------------------|--|
| 2232 (44%) | Confidentiality, Integrity, Accountability |
| 702 (14%) | Integrity, Accountability |
| 443 (9%) | Confidentiality, Accountability |
| 106 (2%) | Confidentiality, Integrity |
| 104 (2%) | Confidentiality, Identification & Authentication |
| 98 (2%) | Confidentiality, Accountability, Privacy |
| 95 (~2%) | Integrity, Accountability, Privacy |
| 90 (~2%) | Integrity, Identification & Authentication, Accountability |
| 86 (~2%) | Confidentiality, Identification & Authentication, Accountability |
| 83 (~2%) | Confidentiality, Integrity, Privacy |

RQ3: How effectively can security objectives be identified and extracted from selected set of documents?

We use recall and precision as measures to assess effectiveness. Table IV presents the results of running the four classifiers against the six documents using a ten-fold cross validation. Creating a “Combined” ensemble classifier demonstrated a slight performance gain over just using the Weka SMO classifier. The “Combined” classifier uses the results of the k -NN classifier if relatively close sentences were found. Otherwise, the “Combined” classifier uses a majority vote of the three classifiers. The k -NN classifier performed equivalently to the SMO classifier. However, the advantage of k -NN classifier comes into play with using the SD tool in an interactive fashion. The classifier reports the sentences closest to the current sentence under test along with the distance. This allows an analyst to view similar sentences when making choices as to the possible security objectives.

The reported precision of .82 implies that the tool correctly predicted 82% of all the security objectives associated with the sentences it classified. The recall score of .79 means that it found 79% of all of the possible objectives. From an error perspective, the precision score implies that 18% of the identified objectives an analyst examines would be false positives, and 21% of the possible objectives were not found.

RQ4: What similarities (words, phrases, grammatical structure, etc.) exist among security-relevant sentences for each security objective??

Overall, keywords are the primary indicator of security objectives for identification/authentication, availability, and privacy. However, for many confidentiality, integrity, and accountability sentences, the grammatical structure of the sentence is often the same. We use these similarities in

grammatical structure and keywords within the sentences of each security objective to develop a set of context-specific templates for composing security requirements. We discuss the proposed templates in section VII.

Table V presents the top twenty keywords listed for security objective. The set of keywords is very similar for confidentiality, integrity, and accountability objectives. This suggests a noticeable relationship among confidentiality, integrity, and accountability objectives.

TABLE IV. TEN-FOLD CROSS VALIDATION

| Classifier | Precision | Recall | F_1 Measure |
|-------------------|-----------|--------|---------------|
| Naïve Bayes | .66 | .76 | .71 |
| SMO | .81 | .76 | .78 |
| k -NN ($k=1$) | .80 | .76 | .78 |
| Combined | .82 | .79 | .80 |

Keywords “system”, “provide”, and “ability” commonly appear in sentences classified as confidentiality, integrity, and/or accountability. Sentences classified as confidentiality, integrity, and/or accountability often appear in the form: “The system shall provide the ability to <action> <resource>”. For example, “The system should provide the ability to check medications against a list of drugs noted to be ineffective for the patient in the past” [ED]. Since the resource in the example sentence involves access to medications (protected health information), the sentence is classified as implying a confidentiality objective. Likewise, since the sentence involves interacting with protected information, the integrity of the data must be maintained. Finally, since the sentence involves a user accessing protected information, the system should keep track of all users who have accessed the data so that they may be held accountable.

For identification/authentication, top keywords include, “authentication”, “login”, “username”, “user”, “authenticate”, and “identify”. While the structure of sentences for confidentiality, integrity, and accountability share a common grammatical pattern, sentences for identification/authentication share only common keywords that suggest the need to know the identity of a user, or the need to ensure that a user has authenticated into the system so that they can be identified by unique credentials.

Similarly, top keywords for availability include “run”, “availability”, “retain”, “time”, “destroy”, “retention”, and “real-time”. Like identification/authentication, no grammatical pattern exists for availability. Instead, keywords that suggest temporal or data retention/destruction obligations are strong indicators of the presence of an availability security objective.

Top keywords for privacy include “consent”, “phi”, “disclosure”, “purpose”, and “privacy”. Again, no grammatical pattern exists in the classified sentences for this objective. Instead, common keywords that suggest privacy objective include terms that involve a user (patients, in healthcare documents) choosing to give consent, or disclosure of protected information to anyone other than the patient. Disclosure of protected information suggests that a user has consented to disclose given information to a third-party.

TABLE V. TOP 20 KEYWORDS BY SECURITY OBJECTIVES

| Security Objective | Keywords |
|---------------------------------|---|
| Confidentiality | system, provide, ability, patient, result, vler, exam, capture, datum, record, send, display, medication, information, list, requirement, status, consuming, order, complete |
| Integrity | system, provide, ability, vler, exam, send, capture, result, datum, store, consuming, patient, pass, click, pick-list, status, application, element, create, generate |
| Identification & Authentication | authentication, login, mac2002, username, oscar, user, authenticate, identify, cash, identity, myoscar, password, waitlist, log, registration, list2012, regen, uniquely, credentials, valid |
| Availability | run, availability, datum, retain, time, year, nurse, destroy, application, legally, recent, retention, care, maximum, real-time, information, period, destruction, record, historical |
| Accountability | system, ability, provide, vler, exam, result, send, consuming, click, pass, patient, capture, pick-list, datum, application, audit, status, store, record, list |
| Privacy | consent, patient, person, phi, disclosure, purpose, privacy, directive, require, organization, ehrus, law, authorization, information, connect, disclose, healthcare, inform, jurisdiction, collect |

VII. CONTEXT-SPECIFIC TEMPLATES

We have developed a set of context-specific templates to translate individual security objectives for each sentence into a set of concrete functional security requirements. We maintain traceability between the original sentence in the natural language artifact and generated security requirements.

RQ5: What common templates for specifying functional security requirements can be empirically derived from the security-relevant sentences?

We have extracted 19 context-specific templates⁹ based on our analysis. Each template is associated with a particular security objective and identifies the conditions under which the template becomes applicable (e.g., based on the subject, action or resource in the security-relevant sentence). Each template also provides one or more reusable parameterized security requirements that can be filled-in to generate system-specific functional security requirements. The context-specific templates, grouped by security objectives, are named below. Details of the templates are available online.⁹

- *Confidentiality*: *C1*-authorized access; *C2*-during storage; *C3*-during transmission;
- *Integrity*: *I1*- read-type actions; *I2*- write-type actions; *I3*- delete actions; *I4*-unchangeable resources;
- *Availability*: *A1*-availability of data; *A2*-appropriate response time; *A3*-service availability; *A4*-backup and recovery capabilities; *A5*-capacity and performance;
- *Identification & Authentication*: *IA1*-select context for roles; *IA2*-unique accounts; *IA3*-Authentication;
- *Accountability*: *AY1*-log transactions with sensitive data; *AY2*-log authentication events; *AY3*-log system events;
- *Privacy*: *PR1*-usage of personal information;

We list example context-specific templates, along with generated security requirements, in Table VI. Requirements analysts should consider our set of context-specific templates to determine which templates apply to each security-relevant

⁹ A complete list of context-specific templates and labeled documents are available at: <http://go.ncsu.edu/securitydiscoverer/>

sentence in the project documentation. We intend a requirements analyst to first identify security objectives using the SD tool on the given project artifacts before considering the templates. The tool produces a set of security objective annotations for each sentence in the documentation and suggests relevant templates. However if the security objectives are already known, the templates can be used independent of the tool as well. For example, for a sentence that the tool annotates as having an accountability objective (or objective is known a priori), the requirements analyst should consider context-specific templates for accountability (AY1, AY2 or AY3⁹). If the sentence contains a subject acting upon sensitive information, the requirements analyst should compose a total of two security requirements to fulfill the sentence's accountability objective (see Table VI).

However, the newly composed security requirements also contain related security objectives themselves. Consider the generated security requirements for AY1 in Table VI. These requirements suggest an integrity objective to prevent modification of log files (I4). The template for AY1 captures this relationship between accountability and integrity allowing the requirements analyst to consider integrity when identifying the security requirements for accountability. In Section VI, we discussed how security objectives for confidentiality, integrity, and accountability often appeared together in the classifications for over 20% of the sentences. The cross-references in our context-specific templates for composing security requirements also reflect the strong relationships among confidentiality, integrity, and accountability.

For a preliminary evaluation, we selected an example use-case from iTrust electronic health record system¹⁰ and applied our process to generate security requirements based on the sentences in the use-case. We identified 32 additional security requirements based on the analysis of just one of the 60 documented use-cases for the system. We have also conducted a user study to evaluate our process and templates for identifying security requirements [39]. Results indicate that our process supports the requirements engineering effort by considering multiple security objectives and identifying an initial set of candidate security requirements for the system.

¹⁰ <http://agile.csc.ncsu.edu/iTrust/wiki/doku.php>

TABLE VI. EXAMPLE CONTEXT-SPECIFIC SECURITY REQUIREMENTS TEMPLATES

| Security Objective | Security Requirements Templates | | Generated Security Requirements |
|--------------------|---------------------------------|--|--|
| Accountability | AY1 | <p>Logging transactions with sensitive data <i>Given:</i> <subject> = user or role <resource> = sensitive information <action> = create/read/update/delete</p> <p>Add Security Requirements:</p> <ul style="list-style-type: none"> The system shall log every time <subject> [performs the] <action> <on for> <resource>. [see C1, I4] At a minimum, the system shall capture the following information for the log entry: <subject> identification, timestamp, <action>, <resource>, and identification of the owner of <resource>. [see IA2, C1, I4] | <p>Input Sentence: The system should provide the ability to check medications against a list of drugs noted to be ineffective for the patient in the past.</p> <p>Security Requirements:</p> <ul style="list-style-type: none"> The system shall log every time user checks medications against a list of drugs noted to be ineffective for the patient in the past. At a minimum, the system shall capture the following information for the log entry: user identification, timestamp, check medication, patient identification. The system shall not allow modification of the log by any user.* |
| Integrity | I4 | <p>Maintaining integrity of unchangeable resources <i>Given:</i> <resource> = write-once information (e.g., log files)</p> <p>Add Security Requirements:</p> <ul style="list-style-type: none"> The system shall not allow modification of <resource> by any user. [see AY1] | |

* Last requirement is generated based on the template for integrity (I4) as suggested in template AY1. Templates for confidentiality can also be considered.

VIII. THREATS TO VALIDITY

We have considered following threats to validity:

Selection of problem domain: Study oracle created using documents from healthcare domain may not be generalizable to other domains due to different security objectives and domain-specific vocabulary. Moreover, assets that need to be protected are well understood in healthcare domain that may facilitate identification of security-relevant sentences. Many organizations adopt data classification guides that can be used to help guide our process in other domains.

Selection of systems and documents: Security requirements may come from different sources (requirements documents, policy specifications, legislative texts, standards and best practices). Variations may exist between security requirements of software systems, even in the same domain. Thus, selection of documents may influence the type and frequency of identified security-relevant sentences.

Selection of security objectives: We have compiled a list of security objectives based on various taxonomies. Our list of security objectives may not be complete. To minimize this threat, we have considered multiple sources from security literature to identify the objectives. A general consensus on the categorization of security objectives minimizes this threat.

Subjective assessment of security objectives: To develop the study oracle, we carried out manual classification of sentences, which can be subjective. Misclassification of sentences based on security objectives in the oracle may have occurred. To minimize this concern, two researchers independently carried out the classification of each document while a third researcher consolidated the final classification. Inter-rater reliability ranges between 0.32 to 0.85, lending validity to the process.

IX. CONCLUSION AND FUTURE WORK

Our work describes a tool-assisted process for identifying key attributes of sentences to be used in security-related analysis and specification of functional security requirements using a set of context-specific templates. We have evaluated

our process on six documents from the electronic healthcare domain, identifying 46% of sentences as implicitly or explicitly related to security. Our classification approach identified security objectives with a precision of .82 and recall of .79. From our total set of classified sentences, we extracted 19 context-specific templates and associated reusable functional security requirements. We also provide an oracle of sentences labeled with relevant security objectives for the healthcare domain¹¹.

To improve the recall of our classification approach and identify security-relevant sentences that may have been missed, we plan to consider features specific to each security objective that may support the classification effort. For instance, we are looking to extract tuples from input sentences that can be used to implement access control. Presence of these tuples can inform the classification for confidentiality and accountability. Identification of such features will also support development of security requirements patterns [40], extending our initial set of context-specific templates. We also plan to evaluate the applicability of our process in domains other than healthcare.

For practitioners, our research can help mitigate security vulnerabilities early in the software development lifecycle by identifying key security requirements that are hidden in plain sight and may otherwise be overlooked.

ACKNOWLEDGEMENT

This work is supported by the USA National Security Agency (NSA) Science of Security Lablet. Any opinions expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSA. We would like to thank the North Carolina State University Ralsearch group for their helpful comments on the paper.

REFERENCES

- [1] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt, "A comparison of security requirements engineering methods,"

¹¹ <http://go.ncsu.edu/securitydiscoverer/>

- Requirements Engineering - Special Issue on RE'09: Security Requirements Engineering*, vol. 15, pp. 7-40, 2010.
- [2] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, "A systematic review of security requirements engineering," *Computer Standards and Interfaces*, vol. 32, p. 13, Jun. 2010.
- [3] J. Slankas and L. Williams, "Automated extraction of non-functional requirements in available documentation," in *International Conference on Software Engineering (ICSE) 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*, 2013, pp. 9-16.
- [4] D. G. Firesmith, "Engineering security requirements," *J. Object Technology*, vol. 2, p. 16, Jan-Feb. 2003.
- [5] D. G. Firesmith, "Specifying reusable security requirements," *Jurnal of Object Technology*, vol. 3, p. 15, Jan-Feb. 2004.
- [6] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Computer Standards and Interfaces*, vol. 29, pp. 244-253, Feb 2007.
- [7] M. Schumacher, E. Fernandez-Buglioni, D. Hyberston, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*. West Sussex: John Wiley & Sons, Ltd, 2006.
- [8] 2012, Common Criteria for Information Technology Security Evaluation, Version 3.1. Release 4. Available: <http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf>
- [9] D. Firesmith, "Specifying reusable security requirements," *Jurnal of Object Technology*, vol. 3, p. 15, Jan-Feb. 2004.
- [10] D. Mellado, C. Blanco, L. E. Sánchez, and E. Fernández-Medina, "A systematic review of security requirements engineering," *Computer Standards & Interfaces*, vol. 32, pp. 153-165, 2010.
- [11] M. Howard and S. Lipner, *The Security Development Lifecycle*. Redmond, WA: Microsoft Press, 2006.
- [12] N. R. Mead, E. D. Houg, and T. R. Stehney, "Security Quality Requirements Engineering (SQUARE) Methodology," Software Engineering Inst., Carnegie Mellon University 2005.
- [13] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, p. 12, 2005.
- [14] A. v. Lamsweerde, "Elaborating security requirements by construction of intentional anti-Models," presented at the International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland, 2004.
- [15] V. N. L. Franqueira, T. T. Tun, Y. Yu, R. Wieringa, and B. Nuseibeh, "Risk and argument: A risk-based argumentation method for practical security," presented at the IEEE International Requirements Engineering Conference, 2011.
- [16] P. Salini and S. Kanmani, "Survey and analysis on security requirements engineering," *Computers and Electrical Engineering*, vol. 38, p. 13, 2012.
- [17] Kurt Schneider, Eric Knauss, Siv Houmb, Shareeful Islam, and J. Jürjens, "Enhancing security requirements engineering by organizational learning," *Requirements Engineering*, vol. 17, pp. 35-56, 2012.
- [18] T. D. Breaux and A. I. Antón, "Analyzing regulatory rules for privacy and security requirements," *IEEE Transactions on Software Engineering*, vol. 34, pp. 5-20, Jan. 2008.
- [19] J. C. Maxwell, A. I. Antón, and P. Swire, "A legal cross-references taxonomy for identifying conflicting software requirements," in *Requirements Engineering*, 2011, pp. 197-206.
- [20] Q. He and A. I. Antón, "Requirements-based Access Control Analysis and Policy Specification (ReCAPS)," *Information and Software Technology*, vol. 51, pp. 993-1009, 2009.
- [21] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie, "Automated extraction of security policies from natural-language software documents," in *International Symposium on the Foundations of Software Engineering (FSE)*, ed. Raleigh, North Carolina, USA, 2012.
- [22] G. Salton and M. J. McGill, "Introduction to Modern Information Retrieval," 1986.
- [23] J. Cleland-Huang, R. Settimi, and P. Solc, "The detection and classification of non-functional requirements with application to early aspects," in *14th IEEE International Requirements Engineering Conference (RE'06)*, IEEE, 2006, pp. 39-48.
- [24] A. Casamayor, D. Godoy, and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," *Information and Software Technology*, vol. 52, pp. 436-445, 2010.
- [25] W. Zhang, Y. Yang, Q. Wang, and F. Shu, "An empirical study on classification of non-functional requirements," in *The Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, 2011, pp. 190-195.
- [26] 2013, Security and Privacy Controls for Federal Information Systems and Organizations. Available: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf>
- [27] 2001, Underlying Technical Models for Information Technology Security. Available: <http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf>
- [28] 2004, Standards for Security Categorization of Federal Information and Information Systems. Available: <http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf>
- [29] 2006, Minimum Security Requirements for Federal Information and Information Systems. Available: <http://csrc.nist.gov/publications/fips/fips200/FIPS-200-final-march.pdf>
- [30] 2002, Federal Information Security Management Act. Available: <http://csrc.nist.gov/drivers/documents/FISMA-final.pdf>
- [31] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Communication of the ACM*, vol. 17, 1974.
- [32] 1990, IEEE Standard Glossary of Software Engineering Terminology. Available: <http://standards.ieee.org/findstds/standard/610.12-1990.html>
- [33] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707-710, 1966.
- [34] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, pp. 159-174, 1977.
- [35] M. Hall, H. National, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software : An Update," *SIGKDD Explorations*, vol. 11, pp. 10-18, 2009.
- [36] J. Han, M. Kamber, and J. Pei, "Data Mining: Concepts and Techniques," p. 744, 2011.
- [37] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, 1986.
- [38] Y. Yang and P. J.P., "A comparative study on feature selection in text categorization " in *Fourteenth International Conference on Machine Learning (ICML'97)*, 1997, pp. 412-420.
- [39] M. Riaz, J. Slankas, J. King, and L. Williams, "Using templates to elicit implied security requirements from functional requirements – A controlled experiment," to present at the International Symposium on Empirical Software Engineering and Measurement (ESEM), Torino, Italy, 2014.
- [40] S. Withall, *Software Requirement Patterns*.: O'Reilly, 2007.

Managing Security Requirements Patterns using Feature Diagram Hierarchies

Rocky Slavin¹, Jean-Michel Lehker¹, Jianwei Niu¹, Travis D. Breaux²

Department of Computer Science¹
University of Texas at San Antonio
San Antonio, Texas, USA
rocky.l.slavin@ieee.org, rpl599@my.utsa.edu,
jianwei.niu@utsa.edu

Institute for Software Research²
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
breaux@cs.cmu.edu

Abstract—Security requirements patterns represent reusable security practices that software engineers can apply to improve security in their system. Reusing best practices that others have employed could have a number of benefits, such as decreasing the time spent in the requirements elicitation process or improving the quality of the product by reducing product failure risk. Pattern selection can be difficult due to the diversity of applicable patterns from which an analyst has to choose. The challenge is that identifying the most appropriate pattern for a situation can be cumbersome and time-consuming. We propose a new method that combines an inquiry-cycle based approach with the feature diagram notation to review only relevant patterns and quickly select the most appropriate patterns for the situation. Similar to patterns themselves, our approach captures expert knowledge to relate patterns based on decisions made by the pattern user. The resulting pattern hierarchies allow users to be guided through these decisions by questions, which introduce related patterns in order to help the pattern user select the most appropriate patterns for their situation, thus resulting in better requirement generation. We evaluate our approach using access control patterns in a pattern user study.

Index Terms—Security, requirements, patterns, feature diagram.

I. INTRODUCTION

A *requirements pattern* is a structure that engineers can use to generate one or more requirements for a recurring situation [1]. Based on design patterns [2], each requirements pattern describes a recurring problem as well as a core solution which can be used repeatedly, but not necessarily in the same way every time. Because situations vary, a requirements pattern must be parameterized to selectively control for those effects that vary across problem spaces. For example, an engineer may choose between single sign-on [3], where users need only one username and password for logging into different systems, as means to maximize usability, where they prefer more complex role-based access control (RBAC) [4] requirements to maximize confidentiality by compartmentalizing access to information. In this example, the engineer perceives constraints differently and desires different levels of control: more control over different systems to manage logins, or more control over classification of resources into roles.

Requirements patterns incorporate engineering knowledge into the solution space of a pattern in order to provide a basis for requirements elicitation and generation. For example, patterns can itemize pre-conditions to indicate when a pattern

applies to a given scenario, or questions whose answers direct the engineer from one solution to another (e.g., from single sign-on to RBAC). Such knowledge reuse allows software engineers to solve problems in a more effective manner. That is, patterns consist of tried-and-tested solutions that have been shown to be effective when applied in the correct context. Consequently, patterns serve as a common language for software engineers to document their design decisions [5].

Security requirements patterns are a special case of requirements that address security risks in a system. Historically, security is dealt with using a penetrate-and-patch approach [6], wherein security problems are identified and addressed in response to penetration testing of the fully-functional system, sometimes in a post-deployment situation. If the system failures are intrinsic to the design, then significant rework is required [7]. Alternatively, it is more cost effective to identify security flaws early in the requirements and design stages of development, which is the focus and intent of security requirements patterns [8].

In the wild, security requirements patterns appear mostly in isolation, either in small sets of related patterns or repositories and related only by a common topic or theme [9, 10]. Combined with the lack of guidance for pattern selection [11], engineers lack the structure needed to holistically address security and balance complex forces or quality attributes (e.g., usability and confidentiality). As the number of security patterns continues to grow [12], engineers face an increased challenge in recognizing what patterns to select [13]. The contribution of our approach is as follows: (1) we propose a new security requirements pattern format that aims to organize requirements knowledge into a canonical form that itemizes this knowledge into inter-dependent questions that investigate the problem space; (2) we demonstrate that these patterns can be linked into a hierarchy to make problem space trade-offs more salient and to connect related patterns to comprise holistic solution spaces; and (3) we evaluate this new format and hierarchy in a user study to measure speed and correctness in selecting patterns to apply to example scenarios.

We evaluate our approach in a user study consisting of graduate and undergraduate students at the University of Texas at San Antonio (UTSA). The study examined our method's ability to deduce the most appropriate set of patterns by having novice users with limited security knowledge and experience use a pattern hierarchy to select patterns for a scenario. We then compared the results with selections that experts chose for

the same scenario. Based on our analysis, we found that novice users not only were more accurate in their pattern selections, but were able to select patterns more quickly than users without a hierarchy. From these results and further observations we assert that the use of pattern hierarchies can improve the usability of security requirements by providing a means for easier access and faster selection as well as better documentation of related patterns.

The remainder of this paper is organized as follows: in Section II, we review the theoretical foundations and background upon which we based our approach; in Section III, introduce our requirements pattern template and in Section IV, we present our pattern hierarchy; in Section V, we present our empirical study design with our results and observations presented in Section VI; finally, we present related work in Section VII, with future work in Section VIII and our conclusion in Section IX.

II. THEORETICAL FOUNDATIONS

Our approach is based on prior work in pattern languages, feature diagrams and the requirements elicitation, which we now discuss.

A. Pattern Languages

Alexander et al. introduced the earliest notion of pattern as a structured device for reusing knowledge in seminal work on building architecture patterns [2]. More recently, there has been substantial work on object-oriented design patterns [14], requirements patterns [9, 15] and security patterns [10, 12, 16]. A security requirements pattern provides a software engineer with a reusable set of requirements to solve common security problems. To be effective, a requirements pattern should incorporate an engineer's knowledge about their system context to select the most appropriate requirement. For example, many systems use passwords to restrict access to sensitive resources (e.g., healthcare data, bank account data, and purchase histories) as well as to associate unique identifiers with individual users (e.g., forums or social networking sites). The risk to individual users of having their password retrieved through a security attack is different depending on the harms of a data breach. A breach to a bank account could be more expensive than the cost of a breach to web forum, where the attacker could at best post derogatory comments. Therefore, an engineer needs a means to tailor reusable knowledge to their situation to yield requirements that balance complex and sometimes competing forces (e.g., performance, security, usability, etc.). Despite this need, current security patterns are not configurable or linkable to enable engineers to tailor security requirements to their needs.

B. Feature Diagrams

A *feature diagram* is a graphical notation to describe how products can be composed from multiple features [17]. A *feature* is an increment of functionality, usually with a coherent purpose [18]. Features may be linked together in a feature diagram to describe common and variable requirements in a system composition [19]. The links between features may be mandatory or optional, and groups of features can be organized using logical connectives such as inclusive or exclusive or.

Figure 1 shows an example including the most common notations used in a feature diagram as well as their meanings.

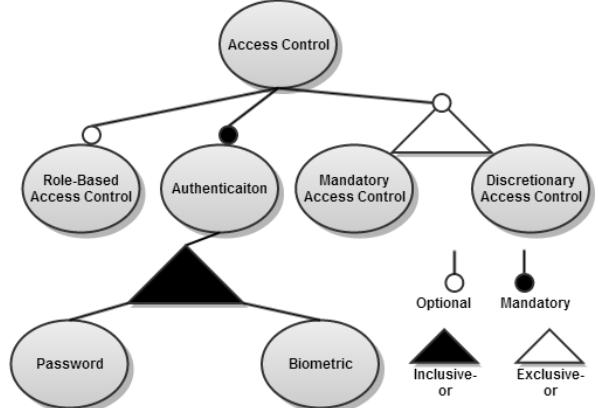


Fig. 1. Feature diagram notation.

Figure 1 describes an example. If we were to describe access control requirements, a mandatory feature would be authentication [12]. An optional feature could be the inclusion of a role-based access control policy. We can further break down some features into groups of features. Here, we can make a decision between the optional mandatory access control (MAC) [20] and discretionary access control (DAC) [21]. We could choose one or the other, but not both, denoted by exclusive-or. Similarly, we must choose between using passwords or biometrics for authentication. In this case, inclusive-or, it is possible to have both, but at least one must be included. It is important to note that the resulting hierarchy does not represent implementation flow. Instead, child nodes represent patterns to be considered if the parent is used.

We felt that feature diagrams would be a suitable notation for visually representing pattern hierarchies because security concepts have mandatory and optional features. By breaking down the features of security concepts, we can construct a diagram using logical connectives which can help a software engineer understand which features are most important for their specific application. Once features associated with the security concept are identified, requirements generation can be customized to the user's specific situation.

C. Inquiry-Cycle

The *Inquiry Cycle Model* (ICM) by Potts et al. [22] is a requirements elicitation method for refining requirements by generating discussion through questions. This model is used to analyze requirements of a system through questions and answers which describe the solutions. We adapt this idea of using ICM questions by using the answers to "how", "why", "what", and "when" questions to direct users to the requirements pattern that will provide them with the resources needed to generate complete requirements. If a user was applying access control, they would start with the Access Control pattern and answer a question about how access to a resource is decided. If the user answers something like "the resource's owner decides who should have access to it," they would find the DAC pattern most suitable. Their answer to this question helped them select the correct type of pattern for their needs. This is a very simple example and not all situations will be so clearly outlined.

The ICM assists in the creation of pattern hierarchies to help users map their needs to relevant patterns in the context of their analysis. Section V describes how we were able to build and refine our hierarchy by interviewing security experts with the ICM. We encouraged experts to ask questions when dealing with hypothetical situations involving security. The experts continued to refine the requirements for the scenarios by asking questions for clarification. By “forcing” them to use an approach similar to the ICM, we were able to elicit the kinds of questions that would be used to connect patterns within hierarchies.

III. PATTERN TEMPLATE

The presentation of pattern content varies depending on the pattern author. In requirements engineering formal and semi-formal methods, such as Tropos [23] and Problem Frames [24], structure requirements knowledge into pattern-like representations. However, many approaches that principally identify themselves as patterns use natural language templates (NLT) to illustrate patterns [25, 26]. These NLT approaches tend to base their template on the so-called *Gang-of-Four* (GoF) [14] book which outlines the following essential pattern elements:

- *Pattern Name* – A simple phrase to describe the problem
- *Problem* – A description of when to apply the pattern
- *Solution* – A general arrangement of the elements used to solve the problem
- *Consequence* – The results and trade-offs of applying the pattern

Our approach includes two additional important features: *forces*, which determine quality attributes that are impacted (maximized, minimized, etc.) by the pattern, and *relations* or links to other patterns using an inquiry-based approach. For example, the access control requirements pattern may contain such forces as generalizability, flexibility, and modifiability since they would be particularly relevant to its application and a question such as, “How are authorization privileges bound to actors and resources?” may lead to an authentication pattern.

By balancing forces using questions which invoke discussion and refinement of security requirements, we aim to minimize the negative consequences or liabilities of the pattern (e.g., access control may require many rules and increase the system’s complexity) as well as capture the applicability of the pattern. The use of an approach based on the ICM provides a way to connect existing security requirements patterns as well as refine system requirements themselves through questions that surface hidden assumptions about the system and its environment. The approach works on two levels: (Level 1) questions relevant to the problem can either draw out specific information from the situation which can be applied to generic requirements in the template in order to generate specific requirements, or (Level 2) they relate the pattern to other patterns relevant to the situation as a means to increase requirements coverage by identifying dependencies or to introduce alternative requirements that balance forces in alternative ways that may be better suited to a particular problem or stakeholder needs. Pattern hierarchies are constructed using the second level, which further supports eliminating unnecessary or unfitting requirements to select the most appropriate pattern.

To catalog patterns, we used a standardized pattern template consisting of six elements. The template we used is an elaboration of ideas devised at RePa’12 and our previous work on the Standardized Pattern Format for deriving characterizations and boundaries of patterns [27]. Using this pattern template is what enables us to construct a pattern hierarchy. To illustrate the different aspects of the template, we use access control as an example.

The template consists of the following elements:

- *Name* – a unique name that is limited to the scope of the pattern
- *Problem* – a statement of the problem to be addressed or a high-level goal to be achieved
- *Context* – domain assumptions that must be true in conjunction with the generated requirements
- *Forces* – the non-functional quality attributes that create trade-offs for the application of the pattern
- *Solution* – a set of questions which refine requirements and guide the pattern user to related patterns
- *Management* – additional information related to the pattern including examples and known uses

We now describe the use of each of the attributes using access control.

A. Name

For our running example, we name the pattern “Access Control” to describe what the pattern covers. In addition, it can be uniquely incorporated into the pattern hierarchy.

B. Problem

The problem shall be expressed either as the security objectives that need to be achieved or the threats that must be mitigated. For our example, we define the problem as “the confidentiality and integrity of resources shall be protected by regulating access to the resources based on different factors.” Here, confidentiality and integrity, [28], two fundamental security attributes, are addressed.

C. Context

A pattern addresses a generic problem in a specific context [12], which shall describe the nature of the situation. This includes any domain assumptions as well as expectations of the system and its environment. For access control we describe the context as “any computer-driven environment in which the access to the resources needs to be regulated” and we list the following assumptions:

- The administrator involved in implementing the authorization system shall be trusted.
- The actors involved in granting or denying authorization shall have the ability to restrict access to the resources being protected.
- Actors shall not assume the identity of other actors with different rights.

We also include one expectation:

- Actors will not circumvent the system to gain access to resources.

D. Forces

Lists of forces are useful for identifying and balancing potentially conflicting or complementary aspects of a system that are imposed by the environment or the requirements [29].

For access control we list three forces:

- Generalizability – The authorization structure must be independent of the type of resources.
- Flexibility – The authorization structure shall be flexible enough to accommodate different types of principals (users or subsystems), objects, and rights.
- Modifiability – It should be easy to modify the rights of a principal in response to changes in his or her or its duties or responsibilities.

E. Solution

To solve the problem and balance the forces, the solution includes a series of ICM-inspired questions that help elicit responses from a user or a subject matter expert. A set of requirements templates with variable parts may be configured by the developer and may correspond directly to the questions. Answers to questions may be pre-conditions to additional questions, pieces to fill in the requirements templates, or guides to other patterns. The questions for our example are:

- Which entities (principals) exist in the system and what resources do they need to access?
- Can two entities access the same resource at the same time?
- How are the resources intended to be accessible?
- Can users of the system be categorized into roles that will have different access privileges? *See Role-Based Access Control pattern.*
- Can resources be assigned labels by the system so users can be given clearance to access resources based on levels of clearance? *See Mandatory Access Control pattern.*
- Can access to resources be regulated by the owner of the resource? *See Discretionary Access Control pattern.*
- How are entities to be authenticated in order to gain access to the system? *See Authentication pattern.*

The requirements template for our example consists of:

- <list of entities> shall <be permitted | not be permitted> to access <resource> simultaneously.
- <list of entities> shall access <resource> through the use of <action>.
- An entity shall be granted or denied privileges to <resource> based on <authorization criteria>

Italicized text refers a pattern-user to a corresponding pattern for the question (as seen in Figure 1). The template also lists generic requirements which can be customized based on the answers to some of the questions. For these requirements, words surrounded by “<” and “>” are mandatory and words surrounded by “[” and “]” are optional. In some cases, a choice of possible answers are provided separated by the “|” character.

F. Management

Finally, a pattern should include any information regarding the source of the pattern, the version, known uses, or any

information relating to how the pattern was derived. For this pattern we note that it was adapted from the security design pattern from [12] and modified for requirements based on Withall’s work on access control requirements [9].

IV. PATTERN HIERARCHY

Pattern relations are managed with the use of feature diagrams, which connect patterns in places where a question might lead a user to another pattern. The resulting hierarchy partitions a larger problem or security area into smaller problems (i.e. patterns). We believe such a hierarchical structure would reduce the time and difficulty needed to select an appropriate pattern and security requirements for a complex situation. Furthermore, the hierarchy enables users to easily discover related patterns which could be implemented to enhance their system security. Feature diagrams are not necessary for the end use of pattern hierarchies, as apparent in our user study. The diagrams serve as a means for visualizing and managing the structure. When knowledge of the feature diagram notation is not present, users can simply be guided with the pattern questions as long as the hierarchy has been constructed correctly.

Based on the solution space of the template, a pattern hierarchy can be derived as seen in Figure 1. The hierarchical relations between patterns appear within each pattern template as the questions that help the pattern user make decisions on which pattern to include. For the Access Control pattern in Section III, the question, “how are entities to be authenticated in order to gain access to the system?” is associated with the Authentication pattern. Figure 1 shows this connection with a link between the Access Control and Authentication patterns. Our approach uses inquiry-based discussion between stakeholders in order to revise previous iterations of requirements and introduce relevant patterns. Our approach begins by highlighting existing challenges for the problem (i.e. the forces) and uses questions to elicit security requirements as responses to those challenges.

We continue to use access control as an example to illustrate a hierarchy due to its large amount of related work including various aspects and specialized models [30, 31, 32]. Not only is it a commonly understood aspect of security, but it also includes many facets which allow us to demonstrate the different benefits of using feature diagrams. In Section III we described a single Access Control pattern from which we extend our example hierarchy as seen in Figure 1. The following subsections describe the use of each notation for our hierarchy design using the same example.

A. Optional

The notation for optional components is denoted by an unfilled circle on the end of the connection closest to the optional pattern. Such components are directly related to the inquiry-based approach. In Section III-E, our example included the question, “Can users of the system be categorized into roles that will have different access privileges?” which related the Access Control pattern to the Role-Based Access Control pattern. For such an instance where the answer to the question determines whether or not the related pattern is relevant, an optional connection would be made.

B. Mandatory

Mandatory components are denoted by a filled circle on the end of a connection closest to the mandatory pattern. In Figure 1, the Access Control pattern includes the Authentication pattern which is commonly viewed as a necessary component of access control [9] as a mandatory feature. Mandatory components do not stem from “yes” or “no” questions as such questions would imply a non-compulsory relation. Instead, such relations are made through “how” questions. For instance, we asked, “*How* are actors to be authenticated in order to gain access to the system?” Here, the word “*how*” poses the question so that it requires an answer. On the contrary, if the question were posed, “*Are* actors to be authenticated in order to gain access to the system?”, one could simply answer “no”, thus avoiding the mandatory component. Here, the phrasing requires that there must be a way for the actors to be authenticated. This question includes a reference to the Authentication pattern and is connected within the diagram as a mandatory component.

C. Inclusive-Or

Some questions may require a child pattern to be included, but allow for multiple children to be included simultaneously. Our example includes two forms of credential verification patterns: Password and Biometric. These patterns are included as an inclusive-or decision since it is acceptable to use multiple credential verification mechanisms and at least one is required. These relations have a filled triangle spanning the edges between the nodes in the decision. Cardinality is represented with brackets in the form $[n, m]$ where n is the lower bound and m is the upper bound. A single number implies an exact amount.

D. Exclusive-Or

Similarly to the inclusive-or case, the exclusive-or relation allows for decisions between different patterns. However, this relation is used when only one choice may be made. These relations have an unfilled triangle spanning the edges between the nodes in the decision. The Access Control node illustrates the use of an exclusive-or decision between the MAC and DAC patterns. An unfilled triangle joins their edges to signify that only one can be chosen if this optional branch is selected.

V. EMPIRICAL STUDY DESIGN

We believe that pattern hierarchies can improve users’ ability to select appropriate patterns and requirements faster and more effectively than unassisted pattern selection. Furthermore, the use of pattern hierarchies should allow novice software engineers with limited experience to benefit from expert knowledge embedded in such patterns and hierarchies.

To evaluate our approach we conducted a study to compare the pattern selection of two groups: one with a pattern hierarchy and one without a pattern hierarchy. The first step was to gather existing security patterns into a repository [33]. This assessment included design, architectural, and requirements patterns so we could better understand how different security concerns were addressed at different development stages among the pattern landscape. Sources included a literature review of textbooks and scientific

publications on security patterns. A total of 143 security patterns, 30 of which we classified as security requirement patterns, were collected. We then gathered a set of patterns relating to access control from the repository and mapped them to the template described in Section III in order to create an initial draft of a pattern hierarchy.

Next, we reached out to security experts from the local security community with the intention of revising our hierarchy and preparing for the evaluation of the hierarchy with novice users. Two banking scenarios [37] regarding access control were presented to these experts and used as a means to gather more information and possible patterns for our existing hierarchy example. From transcriptions of expert interviews conducted using the ICM and feedback directly from the experts, we further refined the Access Control hierarchy. For each of the two scenarios we were also able to assemble a subset of patterns in the hierarchy which were most applicable. These subsets were used later in evaluation with novice users.

Finally, we evaluated the Access Control hierarchy comprised of patterns chosen by the experts by observing what patterns novices selected from it. This evaluation consisted of a comparison of novices both with and without the use of the hierarchy so as to test our hypothesis that the hierarchy would improve users’ ability over unassisted pattern selection.

A. Scenarios

The two scenarios used in our study involved access control in some way due to the common use of access control as an example for security [32]. This allowed for less time to be taken in training since it would be more likely for both experts and novices alike to have some background knowledge in the domain. The scenarios involved a fictional credit union and bank which were partnering with each other. This partnership required new software to be implemented to accommodate shared data. Our description of the entities described the number of employees, their jobs, and features from each of the entities’ existing software infrastructures (e.g., instant messaging system and administrative tools).

Scenario 1 described the ATM system currently in place for both institutions and gave the new requirements for the partnership. These requirements outlined the location, hours of operation, and usage fees for customers of each institution. A general set of auditing requirements were provided as well as the stipulation that only a subset of the bank’s ATMs would be accessible to credit union customers.

Scenario 2 described a computerized banking system with online banking. The system would need to be implemented only for the credit union, but with possibility for relevant data to be transferred to the bank. Specifications pertaining to user access, financial transaction services, customer services, and browser support were described.

These scenario descriptions were intended to include only high-level descriptions of the institutions’ requirements so as to elicit questions and discussion from the experts. This would allow us to infer more about the thought process of the experts interviewed. For situations where clarification on the scenarios was requested, a third-party trained as a domain expert representing the financial institutions was provided. The domain expert explained the different features required for

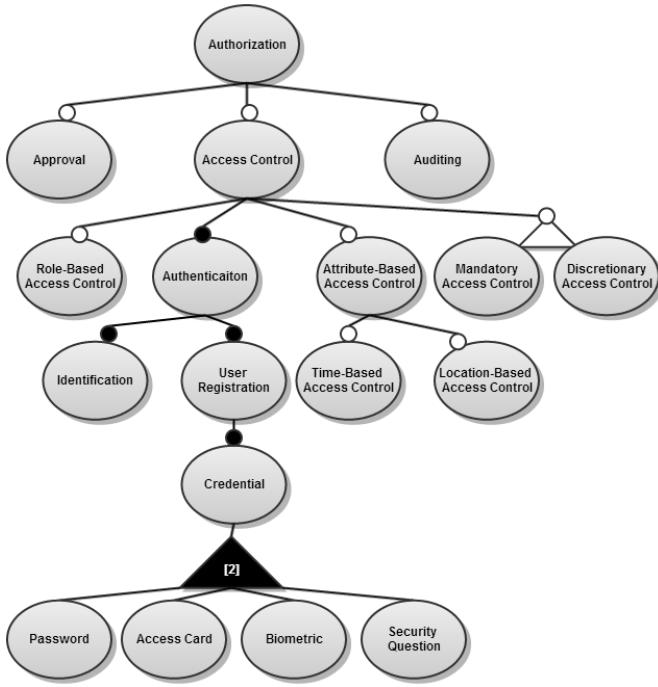


Fig. 2. Access control requirements hierarchy.

employees based on their roles. For any other questions, we required that the domain expert be consistent with his responses so as to provide a constant experience.

B. Sampling

We gathered expert participants with experience relating to computer security from academia and industry. We were able to recruit two members of a local security forum in San Antonio. Members of this forum were industry security specialists in the financial, utility, and local government sectors. We also enlisted participation from a member of a local application security consulting firm, a researcher and professor in the area of access control UTSA, a senior security staff member at a multinational computer technology corporation, and two post-doctoral researchers in the areas of privacy and security policies. In total, we interviewed seven industry and academic experts. Our goal was not to provide a complete representation of the security community's opinion, but instead to create an example to test the use of a hierarchy. For this reason, the representation given by our small sampling of security experts would be sufficient.

The second group consisted of university students with computer science backgrounds. We recruited participants in three undergraduate (junior and senior level) and one graduate-level computer science course. For each class, we gave a three minute explanation of what participants would be asked to do for our study and offered a \$15 gift card as compensation for time. We were able to recruit 34 participants in this way. Due to the course requirements of the department, these students also had some knowledge of computer security. We chose students as subjects due to their fresh entry-level knowledge in access control with little industry experience. This would allow us to see whether pattern hierarchies are useful in relaying expert experience in order to make better and faster decisions.

C. Expert Interviews

We interviewed each expert with an existing hierarchy already created from patterns gathered from textbooks and our pattern repository [9, 12, 33, 35, 36]. Our goal was to see where our hierarchy needed expansion or revision as well as to find a unique subset of the patterns in the hierarchy that would apply to each scenario based on expert opinion. Such patterns subsets would be treated as the gold standard or "correct" set of patterns for evaluating novice performance. Furthermore, by interviewing experts, we sought to better understand how experts decide what security concepts are relevant to a particular system context. This information is fundamental to refining our pattern hierarchy.

The two banking scenarios were presented to each expert at the beginning of the interviews. We asked the participant to assume the role of a software analyst with the task of creating security requirements for the system. The actor playing the domain expert was introduced and the facilitator explained that he would be available for any questions or clarification that may be needed. The same domain expert was present during each interview to answer questions.

Interview questions were planned out according to a script [34] based on the ICM. We used this model as a means to encourage the expert to ask questions that could be used as the pattern-linking questions fundamental to the hierarchy. For example, when expert participants began to consider an access control policy they would ask the domain expert questions such as, "What kinds of users are there?" or "When do you want users to be able to access the system?". We looked for these kinds of cues to construct a hierarchy using the method described in Section IV.

After the interview was completed, we asked the participant to go through the existing list of security requirements patterns in the hierarchy and affirm whether or not the pattern should be regarded for the scenario along with an explanation for inclusion or exclusion. Experts were also encouraged to add anything else they thought was relevant. Transcriptions of the interviews were analyzed along with any notes the experts provided in order to create a final Access Control hierarchy as seen in Figure 2. This resulted in the Authorization pattern [9] being placed at the root of the hierarchy above Access Control. We continue to refer to the hierarchy as an Access Control hierarchy due to our scenarios being focused on access control as well as consistency.

A subset of the patterns in the final Access Control hierarchy was also compiled for each scenario based on expert responses to be used in the novice interviews. To do this, we tallied the selections made by the experts resulting in Figures 3 and 4. If at least five of the seven agreed on a pattern, we included it in the subset of correct patterns for the scenario. This resulted in 12 of 17 patterns for scenario 1 and 13 of 17 for Scenario 2.

D. Novice Interviews

Novice participants were interviewed individually and randomly placed into either a control or an intervention group. Members of both groups were asked to consider the same two scenarios given to the experts and were provided with the same domain expert present during the expert interviews. Participants for both groups were placed into the role of a

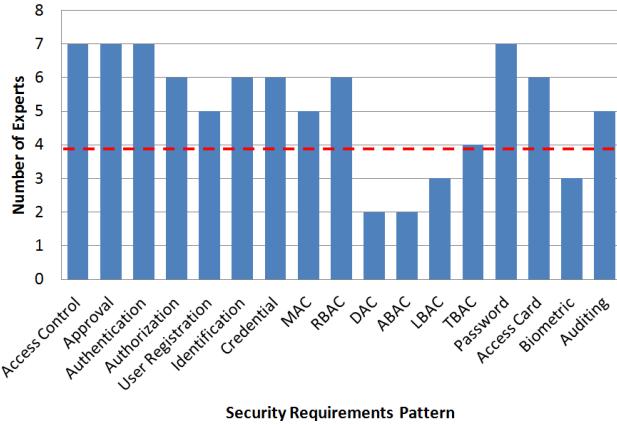


Fig. 3. Scenario 1 expert pattern selection.

software analyst and asked to select the most relevant patterns for each scenario from a list of the patterns from the Access Control hierarchy. Participants were informed that any information they provided would be made anonymous so as to encourage them to proceed naturally. The time to complete both scenarios was also recorded.

For the control group, only the list was given with no relations between the patterns as in the hierarchy. This group was provided with access to the Internet and asked to make the selections to the best of their ability. This gave us a baseline description of how well novices could do in pattern selection on their own.

The intervention group was presented with a questionnaire [34] representing the hierarchy. A questionnaire was used in order to forgo any confusion involving training novices in the use of feature diagrams. The questionnaire was created by organizing the pattern hierarchy questions so that they could correspond to the same checklist of patterns given to the control group. Instructions placed after each question both directed the participants to appropriate consecutive questions and had the participant check off appropriate patterns for the scenario based on the hierarchy.

E. Threats to Validity

Here we discuss both internal and external validity [37].

In order to make statistical comparisons between the two groups, we had to provide the control group with the same set of patterns to choose from as the intervention group. In the wild, it would be up to the user to select from all available patterns. By providing the control group with a list, we were forced to provide them with much of the work that would have been done by the hierarchy. Regarding external validity, we believe that the control group would have performed more poorly in both selection and speed without the provided list. This would actually further validate the use of pattern hierarchies.

We chose not to use the feature diagram representation of the hierarchy for our study due to the time constraints involved for training. This presented a risk to internal validity. Even with the feature diagram notation, the same questions must be answered as in the questionnaire. The feature diagram itself is useful for visualizing the flow between patterns;

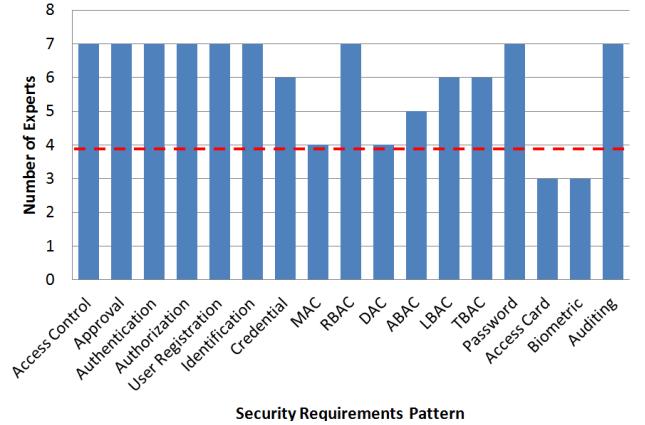


Fig. 4. Scenario 2 expert pattern selection.

however the user must be familiar with the notation. We felt that the questions alone were enough for the user to gain the guidance provided by the hierarchy, and a properly-trained user would have only performed better. By providing the feature diagram representation to participants unfamiliar with it, we would have risked user error due to misunderstanding of the notation.

VI. RESULTS AND OBSERVATIONS

Here we discuss the knowledge we were able to extract from the experts as well as how our hierarchy affected novice decisions.

A. Expert Knowledge Goes Beyond Patterns

An important contribution of patterns is the reuse of expert knowledge. Our research asks if expert knowledge could go beyond individual patterns and expand to pattern organization and selection through the use of pattern hierarchies. Based on our interaction with experts and what we were able to produce with that knowledge, we found that expert knowledge can be applied in a larger scale through such hierarchies.

We used 14 patterns from textbooks and research papers prior to our interaction with experts [9, 12]. From the seven interviews we were able to expand that number to 17 from the additions the experts included to our checklist of patterns. The additions (auditing, time-based access control (TBAC), location-based access control (LBAC), and security question) were not in the form of explicit patterns, but fit into the hierarchy as supplemental security topics [34] related to the existing patterns. We assert that this implies the potential for yet-to-be-created patterns.

The links between patterns which formed the pattern hierarchy were produced by integrating information already present in patterns (e.g., “Related Patterns” sections) and information gained from expert interviews. Generally, as experts described what kind of security requirements were important for the scenarios they would ask the domain expert questions for clarification. For example, when discussing attribute-based access control (ABAC) [38], the ICM urged us to pose the follow-on question: “when should users have access to the system?” Experts would often respond by asking the domain expert if there would be times when employees would not be at work and if they should have access during those times. Depending on the answers to these questions, a

TABLE I. STATISTICAL RESULTS

| | Average Success Rate (%) | | Increase ($\bar{x}_I - \bar{x}_C$) | Standard Deviation (s) | | Sample Size (n) | | Significance (p-value) | Effect Size (Cohen's d) |
|------------|--------------------------|------------------------------|---|------------------------|--------------|-----------------|--------------|---------------------------|----------------------------|
| | Control (\bar{x}_C) | Intervention (\bar{x}_I) | | Control | Intervention | Control | Intervention | | |
| Scenario 1 | 60.78 | 82.92 | 22.13 | 14.70 | 12.41 | 17 | 16 | 0.00003 | 1.63 |
| Scenario 2 | 72.08 | 82.75 | 10.66 | 11.21 | 7.09 | 16 | 17 | 0.002 | 1.16 |

need for a corresponding requirements pattern would be necessary. For this example, the domain expert indicated that there are parts of the day when no employees are at work and thus should not be able to access the system. This triggered the expert to begin considering requirements that would be part of a TBAC pattern. The resulting relation between ABAC and TBAC could now be represented with the question: “are there times when users should not have access to the system?” This is represented in Figure 2 by the optional connection between these two patterns. The question itself would become part of the solution space of the ABAC pattern.

B. Pattern Hierarchies are Efficient and Usable

Novice results were measured by correct pattern choice and time to completion. In both cases, the intervention group performed better than the control group.

Regarding correct pattern choice, novices were graded depending on if they made the same decision to include or exclude a particular pattern as defined by the gold standard pattern sets derived from expert decisions described in Section V-C. Points were not deducted for incorrect answers. A majority of the experts expressed the need for two-factor authentication [39] for both scenarios. To accommodate this, a correct choice of authentication patterns (password, biometric, access card, and security question) consisted of any two. If a participant chose only one of those patterns, they were not awarded points. This selection scheme is apparent in Figure 2 with the range of necessary patterns denoted within the exclusive-or arc with “[2]”.

Before computing the results of our study, we removed outliers by using Iglewicz and Hoaglin’s modified Z-score with median absolute deviation method (*MAD*) [40]. The *MAD* for each group was calculated with:

$$MAD = median(|x_i - \tilde{x}|)$$

Modified Z-scores (M_i) for each score were calculated with:

$$M_i = \frac{0.6755 * (x_i - \tilde{x})}{MAD}$$

Where x_i represents individual scores and \tilde{x} is the median. 0.6755 is the multiplier used for outlier detection recommended by Iglewicz and Hoaglin for samples of our size. For this method, values of M_i greater than $(3.5 * MAD)$ were removed as statistical outliers. This resulted in the removal of two participants.

Table I describes the quantitative results of the study regarding successful pattern choice by novice users. For both Scenario 1 and Scenario 2 novices were more successful at making the same choices as experts when using the pattern hierarchy with a success rate increase over the control group of 22.13% and 10.66% respectively. Regarding the lower increase for Scenario 2, we found that fewer users asked the domain expert questions for that scenario. We took this as an implication of misappropriated familiarity with the situation.

We used a standard t-test to measure the significance of our results. We tested the null hypothesis (H_0) that the mean population success rates for users without the hierarchy (μ_C) and users with the hierarchy (μ_I) were equal:

$$H_0: \mu_C - \mu_I = 0$$

The p-values produced by the t-tests indicate that the null hypothesis could be rejected for both scenarios and that the results of our study were statistically significant.

As an indicator of the strength of our sample size, we calculated effect size using Cohen’s d [41]. This was done by taking the difference between the mean sample success rate for the control group (\bar{x}_C) and the mean sample success rate for the intervention group (\bar{x}_I) divided by the average standard deviation for both groups, s_{IC} :

$$d = \frac{\bar{x}_I - \bar{x}_C}{s_{IC}}$$

While the p-values explained in the previous paragraph give us indications of whether or not our null hypothesis can be rejected, and thus our hypothesis be accepted, effect size is an indicator of whether the result of our experiment is meaningful regardless of our sample size. Based on Cohen’s conventions [41], the values for both Scenario 1 and Scenario 2 indicated a large effect size. This implied meaningful results regardless of our sample size.

Efficiency in terms of completion time was recorded as a total for both groups due to the design of our study. Table II shows an average decrease in selection time of more than 80% for the group using the pattern hierarchy. We attribute this increase in selection speed to the guidance the hierarchy provided. Without the hierarchy, the control group was forced to use their previous knowledge and the Internet to make decisions. Participants were able to make decisions faster while covering the same amount of patterns with the aid of the straightforward questions provided by the hierarchy. We calculated statistical significance and effect size with the same methods used for success rates. Both values implied statistical significance and meaningful results.

VII. RELATED WORK

Researchers have been documenting security patterns for decades, and there have been similar efforts to increase the usability of patterns [16]. We now review and discuss the similarities of related efforts in the security pattern domain, and how they differ from our own work.

Romanosky extends the work done by Yoder and Barcalow [42] by introducing an additional eight security patterns [43]. Romanosky adopts the standardized pattern template originally developed by the GoF, which includes: *Problem*, *Forces*, *Solution*, and *Consequences*. Their template also uses the following additional elements not included in our approach: *Alias*, *Motivation*, *Example*, and *Related Patterns*.

TABLE II. EFFICIENCY

| Average Completion Time (minutes) | | Decrease ($\bar{X}_C - \bar{X}_I$) | Speedup | Standard Deviation (s) | | Sample Size (n) | | Significance (p-value) | Effect Size (Cohen's d) |
|-----------------------------------|------------------------------|---|---------|------------------------|--------------|-----------------|--------------|---------------------------|----------------------------|
| Control (\bar{X}_C) | Intervention (\bar{X}_I) | | | Control | Intervention | Control | Intervention | | |
| 28.56 | 15.82 | 12.74 | 1.81 | 11.85 | 4.95 | 16 | 17 | 0.00037 | 1.52 |

Although many of the element titles are identical in both templates, we assign slightly different meanings. For example, Romanosky's *Solution* section is a textual description of a step or series of steps that, when applied, can mitigate the problem. In contrast, our *Solution* is represented as a series of questions derived using the ICM, to guide users to other patterns and to generate requirements based on answers to these questions as described in Section III. Additionally, questions in Romanosky's approach do not direct readers to related patterns nor do they provide a customized solution based on their answers. Moreover, our *Forces* section is used to influence the types of questions contained within the *Solution*, by surfacing trade-offs that arise from other quality attributes.

Based on the analysis of 220 security patterns, Heyman et al. investigate possible improvements that can be made to increase their usability [44]. Their major conclusions include that the quality of patterns would greatly benefit by the adoption of a common documentation template, and that the construction of a “pattern inventory” would make pattern selection a less daunting task. As opposed to having a general inventory, our approach aims to contextualize each pattern in relation to other relevant patterns through the ICM and Forces. In this way, the pattern user discovers these other patterns as they become relevant and while the pattern user iterates to build a solution from each pattern.

Kienzle and Elder address some of the same problems we address with the security patterns landscape that Heyman et al. documented [45]. Specifically, they address the creation of a common pattern template and pattern repository. Like our template, their common pattern template was derived from the GoF template. Kienzle and Elder augmented the GoF template by including additional security-specific elements. These additions make the patterns somewhat lengthy and, as discussed in the previous paragraph, our *Solution* section provides more than just a textual description of a procedure. Using their template, they have documented a total of 26 security patterns in a patterns repository.

Hafiz et al. introduce an approach for organizing and describing the relationship between patterns using directed acyclic graphs [16]. Their approach uses a hierarchical scheme based on threat models in order to classify security patterns. Patterns which target similar problems are grouped together similarly to our hierarchical approach. With these grouped patterns, the researchers analyzed the order in which they would be applied to describe their relationships and position in the hierarchy. Dotted line arrows and solid line arrows show relationships between patterns for preventing ‘tampering and DoS (denial of service)’ and ‘Escalation of Privilege’, respectively. The authors admit that their pattern language is still a work in progress, is hard to read, and can be “large and intimidating.” Based on the results from our empirical study,

we believe that the use of feature diagrams provides a more usable method of pattern selection for novice users.

Nonfunctional requirements (NFRs) are an important part of requirement specification. Existing work on NFRs defines them as attributes of or constraints on a system [46]. We include forces as a representation of quality attributes in our pattern template. This provides a means for NFRs to be incorporated into the pattern selection process.

The representation and organization of requirements exists in other forms besides feature diagrams. Giorgini et al. use *goal models* to qualitatively relate goals with other goals [47]. This is done with “+” and “-” relationships which signify positive and negative contributions between goals. Similarly to feature diagrams, goal models also incorporate AND and OR compositions. We chose to use feature diagrams due to their expressiveness compared to other goal-oriented modeling techniques which do not allow for the representation of cardinality [48]. Furthermore, the distinction between positively and negatively relating patterns is not necessary for the goal of pattern hierarchies. Hierarchical relationships are strictly positive. Goal models also use binary relations meaning associative compositions are required to create n-ary relations. Feature diagrams include n-ary operations in their most fundamental form.

VIII. CONCLUSION AND FUTURE WORK

Security requirements patterns can greatly reduce the time spent in the requirements elicitation phase of software design as long as the pattern user identifies the correct pattern. Our comparison of pattern selection with and without the hierarchy showed statistically significant and meaningful results. This upholds our hypothesis that pattern hierarchies can be created to allow engineers to make more expert-like decisions in efficient time. Using this method, software engineers can be more confident in the completeness of their requirements specifications. We feel that, based on our results, incorporation of pattern hierarchies with existing patterns can better facilitate knowledge transfer in the requirements engineering domain.

In the future, we plan to modify our digital repository of security patterns to incorporate pattern hierarchies and provide public access in order to both benefit software engineers and gain feedback to further our research. We will also work to mine new patterns for the instances where experts in our case study indicated yet unimplemented patterns (e.g., Auditing and TBAC). Finally, we will work to formalize a method for hierarchy creation including an outline for generating questions used in the hierarchy. In doing so, we hope to provide a means for the creation of more hierarchies in different domains.

ACKNOWLEDGMENT

We thank Daniel Sass for his contribution to the design of our user study and Hanan Hibshi, Maria Riaz, and Laurie Williams for their insightful comments. We thank Daniel Amyot, Xavier Franch and other RePa'12 participants for their

contributions in the initial brainstorming on requirements patterns. This research was funded by Army Research Office (Award #W911NF-09-1-0273). Jianwei Niu and Rocky Slavin have been supported in part by NSF award CNS-0964710.

REFERENCES

- [1] S. Konrad and B. H. Cheng, "Requirements patterns for embedded systems," *RE'02*, pp. 127-136, 2002.
- [2] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, *A pattern language*, Oxford University Press, 1977.
- [3] M. Lin and H. Guo, "Present situation and development of single sign-on technology," *Journal of Computer Applications*, pp. 248-250, 2001.
- [4] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *IEEE Compute* 29(2), pp. 38-47, 1996.
- [5] K. Beck, R. Crocker, G. Meszaros, J. Vlissides, J. O. Coplien, and L. Dominick, "Industrial experience with design patterns," *ICSE'96*, pp. 103-114, 1996.
- [6] G. McGraw, "Testing for security during development: why we should scrap penetrate-and-patch," *IEEE T. Aero. Elec. Sys.* 13(4), 1998.
- [7] B. Boehm, "Software engineering economics," *TSE10*, pp. 4-21, 1984.
- [8] N. Yoshioka, H. Washizaki, and K. Maruyama, "A survey on security patterns," *Progress in Informatics*, No.5, pp. 35-47, 2008.
- [9] S. Withall, *Software Requirements patterns*, Microsoft Press, 2007.
- [10] D. Kienzle, M. C. Elder, D. Tyree, and J. Edwards-Hewitt, "Security patterns repository version 1.0," *DARPA*, Washington DC, 2002.
- [11] M. A. Jalil and S. A. M. Noah, "The difficulties of using design patterns among novices: an exploratory study," *ICCSA'07*, 2007.
- [12] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security patterns: integrating security and systems engineering*, John Wiley & Sons, 2006.
- [13] M. Weiss and H. Mouratidis, "Selecting security patterns that fulfill security requirements," *16th ICSE'08*, pp. 169-172, 2008.
- [14] E. Gamma, R. Helm, R. Johnnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*, Addison-Wesley, 1994.
- [15] T. D. Breaux, H. Hibshi, A. Rao, and J-M. Lehker, "Towards a framework for pattern experimentation: understanding empirical validity in requirements engineering patterns." *RePa'12*, pp. 41-47, 2012.
- [16] M. Hafiz , P. Adamczyk , R. E. Johnson, "Growing a pattern language (for security)," *Onward!'12* pp.139-158, 2012.
- [17] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Carnegie Mellon Univ., CMU/SEI-90-TR-021, 1990.
- [18] P. Zave. (2004). *FAQ sheet on feature interactions* [Online]. Available: www.research.att.com/~pamela/faq.html
- [19] D. Batory, "Feature models, grammars, and propositional formulas," *9th International Conference on Software Product Lines*, pp. 7-20, 2005.
- [20] D. Bell and L. LaPadula. Computer Security Model: Unified Exposition and Multics Interpretation, MITRE Corp., ESD-TR-75-306, 1975.
- [21] G. Graham and P. Denning. "Protection-principles and practice," *AFIPS Spring Joint Computer Conference*, pp. 417-429, 1972.
- [22] C. Potts, K. Takahashi, and A. I. Antón, "Inquiry-based requirements analysis," *IEEE Software*, pp. 21-32, 1994.
- [23] J. Mylopoulos and J. Castro, "Tropos: a framework for requirements-driven software development," *Information Systems Engineering: State of the Art and Research Themes*, pp. 261-273, 2000.
- [24] M. Jackson, *Problem frames; analyzing and structuring software development problems*, Addison-Wesley, 2001.
- [25] C. Palomares, C. Quer, X. Franch, C. Guerlain, and S. Renault, "A catalogue of non-technical requirement patterns," *RePa'12*, pp. 1-6, 2012.
- [26] D. Dietrich and J. M. Atlee, "A pattern for structuring the behavioral requirements of features of an embedded system," *RePa'12*, pp. 1-7, 2012.
- [27] R. Slavin, H. Shen, and J. Niu, "Characterization and Boundaries of Security Requirements patterns," *RePa'12*, pp. 48-53, 2012.
- [28] J. Anderson, "Information security in a multi-user computer environment," in *Advances in Computers* (12), pp. 1-35, 1973.
- [29] S. Lauesen, *Software Requirements: Styles and Techniques*, Pearson Education Limited, 2002.
- [30] R. Crook, D. Ince, and B. Nuseibeh, "On modelling access policies: relating roles to their organisational context," *RE'05*, pp. 157-166, 2005.
- [31] M. Koch, F. Parisi-Presicce, "Formal access control analysis in the software development process," *FMSE'03*, pp. 67-76, 2003.
- [32] H. Hibshi, R. Slavin, J. Niu, and T.D. Breaux, "Rethinking security requirements in RE research," Tech. Rep. Report CS-TR-2014-001, Univ. Texas at San Antonio, 2014.
- [33] J-M. Lehker. (2014). *Security Pattern Repository* [Online]. Available: <http://sefm.cs.utsa.edu/repository/patterns/>
- [34] R. Slavin, J-M Lehker, J. Niu, and T. D. Breaux, "Managing security requirements patterns using feature diagram hierarchies," Tech. Rep. CS-TR-2014-002, Univ. Texas at San Antonio, 2014.
- [35] S. S. Council. Payment Card Industry (PCI) Data Security Standard, 2nd ed., 2010.
- [36] D. Kim, P. Mehta, and P. Gokhale. "Describing access control models as design patterns using roles," *PLoP'06*. 2006.
- [37] R. K. Yin. *Case study research*, 4th ed. In *Applied Social Research Methods Series*, V.5. Sage Publications, 2009.
- [38] L. Wang, D. Wijesekera, S. Jajodia, "A logic-based framework for attribute based access control," *FMSE'04*, pp. 45-55, 2004.
- [39] M.L. Das, "Two-factor user authentication in wireless sensor networks," *TWC'09*, pp. 1086-1090, 2009.
- [40] B. Iglewicz and D. C. Hoaglin (1993), "How to detect and handle outliers", in *The ASQC Basic References in Quality Control: Statistical Techniques*, vol. 16, ASQC Quality Press, 1993.
- [41] J. Cohen, "A power primer," *Psychological Bulletin*, 1992, pp. 155-159, 1992.
- [42] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security," *PLoP'97*, pp. 1-37, 1997.
- [43] S. Romanosky. Security design patterns part 1. <http://citeseer.ist.psu.edu/575199.html>, Nov 2001.
- [44] T. Heyman, K. Yskout, R. Scandariato, and W. Joosen, "An analysis of the security patterns landscape," *SESS'07*. p. 3, 2007.
- [45] D. M. Kienzle and M. C. Elder, "Security patterns for web application development, final technical report", 2003.
- [46] M. Glinz, "On non-functional requirements," *RE'07*, pp. 21-26, 2007.
- [47] P. Giorgini, J. Mylopoulos, and E. Nicchiarelli, "Reasoning with goal models," *ER'02*, pp. 167-181, 2002.
- [48] C. Borba and C. Silva. "A comparison of goal-oriented approaches to model SPLs variability," *ER Workshops '09*, pp. 244-253, 2009.

Engineering Topology Aware Adaptive Security: Preventing Requirements Violations at Runtime

Christos Tsigkanos*, Liliana Pasquale†, Claudio Menghi*, Carlo Ghezzi*, and Bashar Nuseibeh†‡

* Politecnico di Milano, Milano, Italy

† Lero - The Irish Software Engineering Research Centre, Limerick, Ireland

‡ Department of Computing, The Open University, Milton Keynes, UK

Abstract—Adaptive security systems aim to protect critical assets in the face of changes in their operational environment. We have argued that incorporating an explicit representation of the environment’s topology enables reasoning on the location of assets being protected and the proximity of potentially harmful agents. This paper proposes to engineer topology aware adaptive security systems by identifying violations of security requirements that may be caused by topological changes, and selecting a set of security controls that prevent such violations. Our approach focuses on physical topologies; it maintains at runtime a live representation of the topology which is updated when assets or agents move, or when the structure of the physical space is altered. When the topology changes, we look ahead at a subset of the future system states. These states are reachable when the agents move within the physical space. If security requirements can be violated in future system states, a configuration of security controls is proactively applied to prevent the system from reaching those states. Thus, the system continuously adapts to topological stimuli, while maintaining requirements satisfaction. Security requirements are formally expressed using a propositional temporal logic, encoding spatial properties in Computation Tree Logic (CTL). The Ambient Calculus is used to represent the topology of the operational environment - including location of assets and agents - as well as to identify future system states that are reachable from the current one. The approach is demonstrated and evaluated using a substantive example concerned with physical access control.

I. INTRODUCTION

Adaptive security systems aim to protect critical assets in the face of changes in their operational environment. They do so by monitoring and analysing this environment and deploying security controls that satisfy some security requirements. A key characteristic for engineering adaptive security is the *topology* of the operational environment [20] that can denote the structure of a physical space, such as a building, the location of assets and agents in that space and their structural relationships. These relationships may determine whether assets or agents are co-located (proximity), if an agent can access a specific asset or location (reachability), or if an asset, agent or an area is enclosed by another area (containment).

In addition to existing context models [1], topology can provide a system with both structural and semantic awareness of important contextual characteristics that can affect security concerns. Security requirements can be expressed in terms of proximity and reachability relationships among assets and

We acknowledge SFI grant 10/CE/I/1855 and ERC Adavanced Grants (ASAP) no. 291652 and (SMScom) no. 227977.

agents; for example, a security requirement can specify that an asset should never be co-located next to an unauthorised agent who can harm its integrity. The location of agents, who can harm the assets placed in their vicinity raises potential security threats. For example, a threat can arise if a malicious agent can reach a valuable asset from the area in which she is located. Knowing where valuable assets are placed and their relationships to other objects in their proximity is also important in order to identify possible security controls that can be enacted to protect them. For example, authorisation mechanisms may be needed in some of the areas that could be accessed by a malicious agent to harm an asset.

Changes in topology due to movements of assets or agents, or due to changes in the structure of a space can affect system security concerns and determine violations of security requirements. For example, the movement of a valuable asset to a different location may lead to a violation of the asset’s confidentiality because illegitimate users can reach the new asset’s location. Movements of agents and changes in the structure of a space can also bring new threats. In particular, agents’ movements may cause harm to the assets that can be reached from their current location. Moreover, merging two adjacent rooms can make a valuable asset located in one of the rooms reachable by potentially malicious agents that can access the other room.

This paper proposes to engineer adaptive security systems by identifying and preventing violations of security requirements triggered by changes in the topology of the operational environment. To achieve this aim, a live representation of the topology is maintained at runtime and is updated when assets are moved, agents perform actions, or the structure of the space is altered. This allows reasoning on the consequences that topological changes can have on the satisfaction of security requirements at runtime. When the topology changes, we look ahead at a subset of the future system states; these states represent the topological configurations that are reachable when agents perform a sequence of actions. If a security requirement can be violated in one of the future system states, a set of security controls is applied, by revoking from some agents the rights to perform certain actions that lead the system to undesired topological configurations. Security controls are selected by using different criteria, such as whether they satisfy other non-security requirements or minimise the cost (i.e. minimise the number of security controls applied).

This paper focuses on topologies describing the structure of a physical area or a building, in contrast to digital topologies which have been the subject of discussion elsewhere [20]. We propose a systematic model-based software engineering approach to formally reason on system properties and enforce security requirements satisfaction. The Ambient Calculus [6] is used to represent the topology of the operational environment - including location of assets and agents - as well as to identify future topological configurations that are reachable from the current one.

Requirements, expressing desired properties related to access control, are formally expressed using Computation Tree Logic (CTL) [9], where spatial properties are encoded by using a set of atomic propositions. The proposed approach is illustrated and evaluated through a substantive example concerned with physical access control. Our experimental results demonstrate the effectiveness of the approach in selecting security controls that avoid entering topological configurations where security requirements are violated. The number of system states to be analysed at runtime is also greatly reduced by excluding agents' actions and locations that do not influence the satisfaction of security requirements.

The rest of the paper is structured as follows. Section II describes the case study adopted throughout the paper to explain and evaluate our work, and Section III provides an overview of our approach. Section IV introduces the formalisms adopted to represent the topology of the operational environment, the system state space and the system requirements. Section V explains the threat analysis performed to detect potential violations of security requirements. Section VI describes the approach adopted to identify and select security controls. Section VII provides experimental results, Section VIII describes related work, and Section IX concludes the paper.

II. CASE STUDY

As a case study we consider a system regulating access to a building hosting lecture theaters and staff offices. Figure 1 represents the map of the building floor layout in which access to locations (rooms and areas) must be regulated. The building is composed of an entrance area (Bld) and the offices areas (A1 and A2). From the building entrance (Bld) one can access lecture theaters LT1 and LT2, as well as the non-faculty offices area (A1). From A1 it is possible to access the researchers' offices (O1, O2, and O4), the printer room (O3), and the faculty's offices area (A2). From A2, it is possible to access professors' offices (O5 and O7), and a room were a safe is located (O6).

In our scenario, valuable assets need to be protected, such as the safe's security code and a new server that will be positioned in office O2. Agents roaming in the building can for example be postdocs or professors, or external entities such as visiting technicians. The access control system should restrict access to the various areas of the building to pursue the following requirements, which here are informally expressed in terms of structural relationships among assets and agents in the physical space.

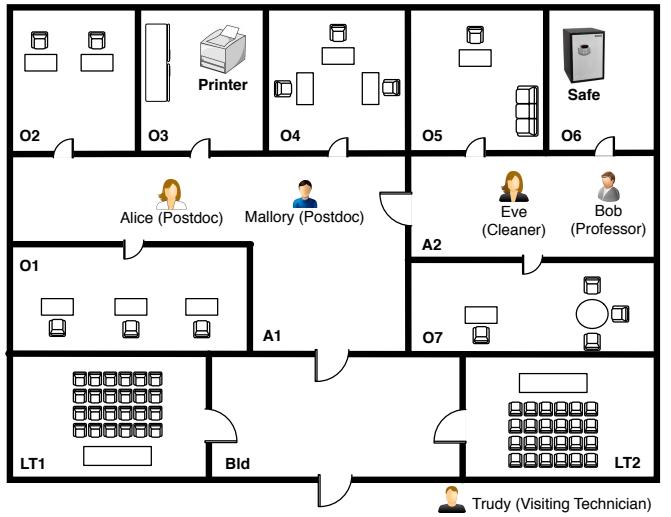


Fig. 1. Map of the academic building used in our case study.

Security Requirements aim to protect valuable assets in the building from damage, theft or improper use. Specifically,

- *SR1*: To guarantee the confidentiality of the safe's security code, this security requirement asserts that no one can be co-located in the safe room (O6) together with one of the agents (e.g., Bob) who knows the safe's security code.
- *SR2*: To preserve the integrity of the server, this security requirement states that only when authorised staff are present (e.g., Alice), other people can be in the same room with the server.
- *SR3*: To prevent harm to persons or assets located in the building, this security requirement claims that an external visitor must always be in specific public areas or in the room where she has to carry out her work. For example, since the visiting technician (Trudy) has to fix the printer located in O3, she can only be in Bld, A1, or O3.

Functional Requirements aim to guarantee that the research centre staff can perform their own work. In particular,

- *FR1*: Alice must be able to access her office O2.
- *FR2*: Bob must be able to access the room O6, which contains the safe.
- *FR3*: The visiting technician must be able to access the room O3, where the printer to be repaired is located.

We envisage different scenarios characterised by topological changes that might require adapting existing authorisation permissions.

Scenario 1. Agents' movements can change the security controls that need to be applied. For example, if a professor (Bob) enters in the safe room, no one else should be allowed to access O6 until the professor exits, in order to satisfy security requirement *SR1*. Similarly, if the cleaner (Eve) is in O6, authorisation to access O6 should be revoked from Bob until Eve exits, in order to guarantee the safe code's confidentiality.

Scenario 2. A new asset (server) is placed in O2 and introduces security requirement *SR2* aiming to guarantee the

server's integrity. In this scenario, when a potentially malicious, unauthorised agent (e.g., Mallory), enters the building authorisation mechanisms must be adapted in order to avoid Mallory being co-located with the server, without the presence of an authorised person (e.g., Alice).

Scenario 3. A technician (Trudy) visits the building to fix the printer and therefore security requirement *SR3* must also be satisfied. In this case, Trudy should be authorised to enter only the areas that have to be traversed to reach O3.

III. TOPOLOGY AWARE ADAPTIVE SECURITY

Topology is the study of shapes and spaces, including properties such as connectedness and boundary. We define topology aware adaptive security as the *adaptation process that aims to continue to satisfy security requirements at runtime, even when the structure of the operational environment changes*. Figure 2 depicts the activities of the MAPE (Monitoring, Analysis, Planning, Execution) loop [13] supported by our approach in order to engineer topology aware adaptive security. These activities rely on a representation of the topology, which is kept in sync with the structure of the physical space and the location of assets and agents in that space. The system security and non-security requirements (respectively S and NS in Figure 2) are also modelled explicitly in order to configure the analysis and planning activities.

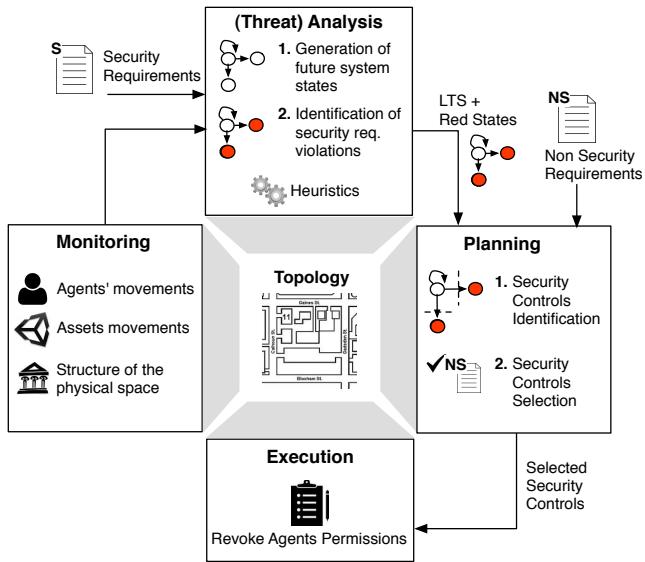


Fig. 2. The MAPE loop to support topology aware adaptive security.

Every time a change in the topology of the operational environment or in the system requirements is observed, a new adaptation cycle through the MAPE loop is triggered. This process is agnostic of previous adaptation cycles and security controls are selected independently each time.

Monitoring captures topological changes that can be determined by agents' movements, assets movements, and alterations of the structure of a physical space, and updates the model of the topology accordingly. Note that assets and

agents' movements can be monitored automatically by using, for example, smart cards. Alterations to the physical space, on the other hand, require the manual intervention of a human agent to be included in the representation of the topology.

Analysis is triggered by topology changes detected during the monitoring activity and discovers potential threats that may cause requirements violations. To achieve this aim, future system states are generated and violations of security requirements are identified. The generation of future system states is based on the look-ahead of subsequent executions of actions (i.e. movements) by the agents within the physical space. Assets movements and modifications in the structure of the physical space are monitored as possible exogenous topological changes that may occur. The threat analysis checks that endogenous actions by agents do not lead to violations of security requirements. States in which a violation takes place are also identified (dark states in Figure 2). We also propose a set of heuristics to reduce the system state space by not considering parts of the topology which are not relevant for the satisfaction of security requirements. Threat analysis can also be employed at design time, when it is more feasible to undertake an exhaustive look-ahead of all possible sequences of actions that agents can perform. However, applying threat analysis only at design time reduces effectiveness of detecting unexpected security threats determined by changes in the topology or in the security requirements arising at runtime.

Planning identifies security controls by detecting a suitable set of actions that have to be forbidden to specific agents in order to avoid security requirements violations. The configuration of security controls that is selected should also satisfy other non security requirements, if possible.

Execution changes existing system authorisations by revoking from agents the right to perform the actions forbidden by the selected configuration of security controls. However, other security controls [10] (e.g., obligation and dispensation) can also be supported.

IV. MODELLING FORMALISMS

This section introduces the Ambient Calculus, Labelled Transition Systems (LTS) and Computation Tree Logic (CTL), which are the formalisms adopted to model the topology of the operational environment, the system state space, and the system requirements, respectively.

A. Ambient Calculus

The Ambient Calculus is a process algebra having a special focus on mobility [6]. An *ambient* is an abstract entity that can model different elements both in a physical space (e.g., agents and locations) and in a digital space (e.g., programming scopes and variables) [17]. Ambients reside in a hierarchy of locations and form a tree structure that can be dynamically re-configured when they exercise a set of capabilities (actions), such as *in*, *out*, and *open*. In this work, a fragment of the Ambient Calculus is considered where the communication primitives and the *open* capability are neglected.

The syntax of the Ambient Calculus used in this paper is described in formulae 1a to 1f, overleaf. A process *P*

can simply do nothing (formula 1b), can be decomposed in two processes running in parallel (formula 1c), or can be enclosed into an ambient which is a particular kind of process (formula 1d). A process can also execute a capability and then proceed to the execution of another process. We assume that the capabilities available are *in* and *out* (formulae 1e and 1f, respectively).

| | | | |
|-----------|---------------------|--------------------------------|------|
| P, Q, R | $::=$ | <i>processes</i> | (1a) |
| | $ 0$ | <i>inactivity</i> | (1b) |
| | $ P \mid Q$ | <i>parallel composition</i> | (1c) |
| | $ n[P]$ | <i>ambient</i> | (1d) |
| | $ \text{in } n.P$ | <i>capability to enter } n</i> | (1e) |
| | $ \text{out } n.P$ | <i>capability to exit } n</i> | (1f) |

An Ambient Calculus formula can represent both the description of the structure of an environment and its evolution. The former expresses how ambients are structured and nested, while the latter describes how the structure of the environment can evolve through the execution of a given set of capabilities.

We represent the topological configuration of the operational environment using Ambient Calculus formulae. The hierarchical relation is exploited to describe how different ambients are nested. We explicitly distinguish among different types of processes representing assets, locations, and agents. For example, the topology of the building described in Section II is encoded as Ambient Calculus formulae 2a-2c¹. Formula 2a denotes that locations LT1, LT2 and A1 are on the same hierarchical level, since to access them an agent must be located in the building entrance (Bld). These formulae also specify the locations of the agents, namely that Trudy is inside Bld, Alice and Mallory are in A1, Eve and Bob are in A2, and the Server and the Safe are located in O2 and O6, respectively.

$$Bld[LT1 \mid LT2 \mid A1 \mid Trudy] \quad (2a)$$

$$A1[Alice \mid Mallory \mid O1 \mid O2[Server] \mid O3 \mid O4 \mid A2] \quad (2b)$$

$$A2[Eve \mid Bob \mid O5 \mid O6[Safe] \mid O7] \quad (2c)$$

We assume that assets and locations cannot perform any capability, while agents can always execute any possible capability depending on their current location. More precisely, an agent can always enter (*in*) all co-located areas and exit (*out*) from the current room or area. For example, in the configuration described in formulae 2a-2c, Trudy can perform *out Bld*, *in LT1*, *in A1* or *in LT2*, while Bob and Eve can perform *out A2*, *in O5*, *in O6*, and *in O7*. Therefore, we do not explicitly represent capabilities into an Ambient Calculus formula, as they can be inferred from how the ambients are structured and from the types of ambients in the topology.

B. Labelled Transition Systems

Labelled Transition System [9] (LTS) is a modelling formalism used to describe systems and their evolution in terms of states and transitions. States usually specify the possible configurations of the system. Transitions describe how the

configuration of the system can change by moving from one state to its successors.

Formally, given a set AP of atomic propositions, a LTS \mathcal{M} is formally described as a tuple $\mathcal{M} = \langle S, S_0, R, L \rangle$, where:

- S is a finite set of states;
- $S_0 \subseteq S$ is the set of initial states;
- $R \subseteq S \times S$ is a transition relation that must be *total*, that is for every state $s \in S$ there is a state $s' \in S$ such that $(s, s') \in R$;
- $L : S \rightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions that are true in that state.

C. From Ambient Calculus to LTS

Starting from an Ambient Calculus formula, interpreting it over an LTS means describing the topological evolution of the system based on execution of capabilities. Each LTS state represents a different topological configuration. Atomic propositions are used to describe where agents and assets are located (spatial modalities). For example, atomic proposition '*Bob in O6*' is true when Bob is located in office O6. LTS transitions connect different states and correspond to the execution of capabilities by agents.

D. Computation Tree Logic

CTL [9] is a branching temporal logic used to specify temporal properties that a system must satisfy. CTL includes two types of formulae: *state* and *path* formulae. State formulae are defined over a set of atomic propositions AP using the grammar specified in 3, where $a \in AP$ and φ is a path formula. A state formula can be an atomic proposition a , the special proposition *true*, the composition (\wedge) of two sub-formulae, the negation (\neg) of a formula, and a path CTL formula prefixed by E (exists) or A (always) path quantifiers. E predicates that φ must hold on at least one path starting from the current state, while A asserts that φ must hold on all paths starting from the current state.

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg \Phi \mid E\varphi \mid A\varphi \quad (3)$$

CTL *path formulae* are defined in 4. A state CTL formula Φ prefixed by the *next* operator (X), and two CTL formulae Φ_1 and Φ_2 linked by the *until* operator (U) are valid CTL path formulae.

$$\varphi ::= X\Phi \mid \Phi_1 U \Phi_2 \quad (4)$$

A state formula Φ is evaluated over a state s of the LTS. For example, property $\Phi = \Phi_1 \wedge \Phi_2$ is true in a state s if and only if s satisfies both Φ_1 and Φ_2 . Similarly, property $E\varphi$ is true in a state s of the LTS iff there exists a path π starting from s that satisfies φ . Path formulae are interpreted over infinite paths of the LTS. For example, given an infinite path π , the property $\Phi_1 U \Phi_2$ is true if there exists a state s in the path that satisfies Φ_2 and each state that precedes s on the path satisfies Φ_1 . The interested reader can refer to [9] for a complete description of the semantics of CTL.

In this paper CTL is used to specify security as well as other requirements. For example, Formulae 5a-5c represent

¹Note that LT1, ..., O7 is used as a shortcut to $LT1[0], \dots, O7[0]$.

security requirements *SR1-SR3*. Variables Y and Z are used to denote (implicitly universally quantified) agents (such as Alice, Mallory, ...) and locations (such as Bld, A1, ...). Moreover, an atomic proposition, such as '*Bob | Eve*' denotes that Bob is co-located with Eve.

$$SR1 : AG(\neg((Bob | Y) \wedge (Bob in O6))) \quad (5a)$$

$$SR2 : AG(\neg((Y | Server) \wedge \neg(Alice | Server))) \quad (5b)$$

$$SR3 : AG(\neg(Trudy in Z)) \quad (5c)$$

Formulae 6a-6d represent security requirement *SR1* by associating Y with Alice, Mallory, Eve, and Trudy.

$$AG(\neg((Bob | Alice) \wedge (Bob in O6))) \wedge \quad (6a)$$

$$AG(\neg((Bob | Mallory) \wedge (Bob in O6))) \wedge \quad (6b)$$

$$AG(\neg((Bob | Eve) \wedge (Bob in O6))) \wedge \quad (6c)$$

$$AG(\neg((Bob | Trudy) \wedge (Bob in O6))) \quad (6d)$$

In this paper CTL is also adopted to express the functional requirements of the case study. In particular, formulae (7a)-(7c) express functional requirements *FR1-FR3*.

$$FR1 : EF(Alice in O2) \quad (7a)$$

$$FR2 : EF(Bob in O6) \quad (7b)$$

$$FR3 : EF(Trudy in O3) \quad (7c)$$

V. THREAT ANALYSIS

Threat analysis aims to predict potential violations of security requirements that can take place in future system states. This section describes the process adopted to generate future system states reachable from the current topology. It also explains how violations of security requirements - that can take place in future states - are detected. The section concludes by illustrating a set of heuristics that can be adopted to reduce the state space analysed.

A. Generation of Future System States

Future system states are generated after changes in the topological configuration of the operational environment take place. Topological changes are continuously monitored and the representation of the topology is updated at runtime accordingly. Any subsequent execution of actions by the agents within the physical space (agents' intentions) is exhaustively considered and the corresponding future system states are generated. This is performed because future agents' actions cannot be predicted in advance. Movements of assets and modifications in the structure of the physical space are not considered, because they are conceived as possible exogenous topological stimuli, and therefore cannot determine requirements violations spontaneously.

Future system states are generated in two distinct steps: (I) exhaustive look-ahead of action executions by agents and (II) LTS generation, which translates the representation of the topological configuration into a formalism that is suitable for verification. Exhaustive (limited scope) look-ahead of actions execution depends on the structure of the physical space and on the current position of agents. Each agent can enter

accessible areas with which she is co-located, and can exit from the area where she is currently placed. However, the exhaustive look-ahead is bounded by the number of steps selected to be considered in the future by the agents. The portion of the state space that is generated has a limited depth. This corresponds to the execution of a predefined number of actions per agent.

We assume a look-ahead bound of one step² (i.e. each agent always performs at most one action per location). In this way, at one step in the future, capabilities that can be performed with respect to all locations accessible directly by agents are examined. For example, in Figure 1, Trudy can perform *in LT1*, *in LT2*, *in A1*, and *out Bld*. Each action performed by an agent at a specific step is considered to be performed in parallel with those of the other agents at the same step, in a way that is consistent with process calculi. Any possible ordering in which capabilities can be executed by the agents is also considered. Even though different sequences of the same capabilities can lead to the same topological configuration, their ordering is relevant because it can affect the satisfaction of security requirements differently. More precisely, the generation of all the possible action interleavings is fundamental to identify a specific sequence of actions that lead to the violation of a security requirement. For example, if both Mallory and Alice enter O2 at the same time, the sequence of actions where Mallory accesses O2 before Alice will determine a violation of security requirement *SR2*. In contrast, the same requirement holds if Alice enters O2 before Mallory.

Algorithm 1 sketches the procedure we use to look ahead at capability executions and to generate all the sequences of actions that agents (A_g) can perform from the current topology T_c . For each agent a (Line 2), the set of relevant movement intentions (Line 3) corresponding to accessing (*in*) co-located locations or exiting (*out*) from the current location are considered. Function f associates each agent with co-located rooms and areas. For each of these locations (Line 4), function *GETACTION()* returns the relevant action that an agent can perform on it (Line 5). The set A_{act} (Line 6) is populated with all the possible capabilities an agent can do depending on the accessible locations ($\{\langle a, action, s \rangle\}$). Function z associates each agent with the possible actions she can execute (Line 8). When multiple agents are present, the product of their potential capabilities is computed (Line 10), to account for different orderings of capability executions. This is equivalent to inserting sets of capabilities into an Ambient Calculus formula representing the current topological configuration. Afterwards, given the current topology T_c , each possible set of interleaving actions is mapped to the transitions of the LTS (Line 11).

To generate the LTS, a set of states is subsequently identified starting from the current one representing the current topology. The procedure used to accomplish this task is the one described by Mardare et al. [16], [15], which uses sets

²One might also consider larger scopes, although the larger the scope the more likely it is that exogenous changes happen and invalidate the analysis.

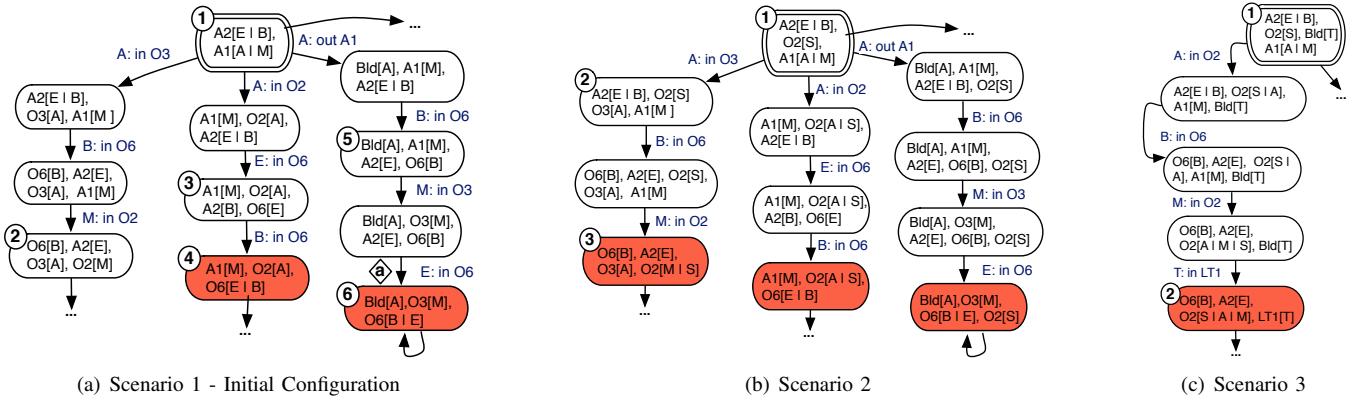


Fig. 3. Fraction of the LTS generated from the case study; violating states are in dark.

Algorithm 1 Exhaustive look-ahead of capability executions

```

1: function LOOK-AHEAD( $T_c, A_g$ )
2:   for  $a \in A_g$  do
3:      $A_{act} = \{\}$ ;
4:     for  $s \in f(a)$  do
5:        $action = GETACTION(a, s)$ ;
6:        $A_{act} = A_{act} \cup \{(a, action, s)\}$ 
7:     end for
8:      $z(a) = A_{act}$ 
9:   end for
10:  for  $ac \in \text{PRODUCT}(z(\cdot))$  do
11:    MAPONLTS( $ac, T_c$ );
12:  end for
13: end function

```

for the representation of state information. Each transition corresponds to the execution of a capability by an agent, while each state of the LTS describes a different topological configuration of the operational environment; note that different states can also be associated with the same topology in case this is reached by performing different sequences of agents' capabilities. Figure 3 partially represents the LTS generated from different topological configurations identified in the scenarios proposed in Section II.

B. Identification of Security Requirements Violations

The LTS representing agents' potential movements is explored to identify those states where security requirements can be violated. Such states are detected by checking the LTS against CTL formulae representing security requirements. Figure 3 highlights in dark the states where security requirements are violated³ in our scenarios. The attack scenarios appearing in a possible evolution of the system can be summarised as follows.

- In the first scenario (Figure 3a), Trudy is not in the building and the server is not placed in O2 (state 1). The first branch (from the left) of the LTS does not cause any violation of security requirements because although Mallory enters O2 the server is still not placed in it. The

action sequences represented by the second and the third branch cause violation of security requirement $SR1$. In the second branch, Eve accesses O6 (state 3) and afterwards Bob accesses the same room (state 4). In the third branch, Bob accesses the safe room (state 5) and subsequently Eve accesses the same room (state 6).

- In the second scenario (Figure 3b), the server (S) is placed in room O2 (state 1). The action sequence represented by the first branch from the left violates security requirement $SR2$. This is because Mallory accesses O2 (state 3), where the server is located, when Alice is not present, since she is in O3 (state 2).
- In the third scenario (Figure 3c), Trudy accesses the building (state 1). An action sequence that may cause a violation of security requirements is when Trudy moves to rooms (e.g., LT1) she is not supposed to traverse to reach the printer room (state 2). This action sequence violates requirement $SR3$.

The theoretical space complexity to identify security requirements violations depends on the number of states and transitions that are generated in the worst case. To compute the number of transitions of the LTS it is necessary to remember that each of them is associated with an action. Given a set of agents A_g and locations L , in the worst case an agent $a_1 \in A_g$, can perform $|L|$ actions that is entering or exiting every location $l \in L$. Since each action performed by a_1 can be combined with every other action performed by another agent, these actions can be aggregated in $|L|^{|A_g|}$ sets of $|A_g|$ actions (one for each agent). It is also necessary to consider the possible permutations of the agents' actions (i.e. all the possible orderings in which actions can be performed). More precisely, a set of $|A|$ actions can be ordered in $|A|!$ ways. Thus, the total number of transitions in the worst case scenario is $|A_g|! \cdot |A_g| \cdot |L|^{|A_g|}$. Since each transition moves the system to a new state, in the worst case scenario the total number of states of the LTS is $|A_g|! \cdot |A_g| \cdot |L|^{|A_g|} + 1$, that is the number of transitions plus the initial state representing the initial topological configuration of the system.

Once a LTS is generated (e.g., \mathcal{M}), the complexity of verifying if a formula φ holds in \mathcal{M} is $\mathcal{O}((|S| + |R|) \cdot |\varphi|)$ [9].

³Self-loops on final states are added to comply with the LTS definition.

Thus, in our case the time complexity is $\mathcal{O}(|A_g|! \cdot |A_g| \cdot |L|^{|A_g|} \cdot |\varphi|)$.

C. State Space Reduction Heuristics

State space reduction heuristics aim to reduce the sequences of actions that are considered, and consequently, the number of LTS states generated. This paper proposes two types of domain-specific techniques to achieve such reduction. Firstly, we use the topology of the operational environment to avoid considering agents' actions that are irrelevant to the satisfaction of security requirements. Secondly, a new type of analysis that is reactive instead of proactive is enacted, when other measures that aim to meet runtime demands are ineffective.

The first heuristic depends on the pattern adopted for the specification of security requirements, which indicates how agents, assets and locations are related to each other. In particular, this heuristic only focuses on the agents' actions that can determine the violation of the specified security requirements. Let us consider security requirements that are expressed as $AG(\neg(a_g \text{ in } l))$, whose violation is determined in those states where agent a_g reaches location l . The look-ahead of the actions performed by agent a_g can therefore focus only on those allowing a_g to reach location l within the number of steps in the future that are analysed. For example, to check security requirement *SR3* assuming the topological configuration where *Trudy* is in *Bld*, it is necessary to consider only the actions that allow *Trudy* to reach accessible locations $l \in Y$, such as *Trudy in LT1* and *Trudy in LT2*.

Consider security requirements that are expressed as $AG(\neg((a_{g1} \mid a_{g2}) \wedge (a_{g2} \text{ in } l)))$, whose violation is determined when two agents (a_{g1} and a_{g2}) are co-located in the same place (l). In this case, the look-ahead can focus only on those actions allowing a_{g1} and a_{g2} to reach location l in one step. For example, to check security requirement *SR2*, assuming the topological configuration shown in Figure 1, look-ahead of agents' actions will only include *Bob in O6* and *Eve in O6*.

Consider security requirements that are expressed as $AG(\neg((a_{g1} \mid a_s) \wedge \neg(a_{g2} \mid a_s)))$, whose violation is determined when an unauthorised agent (a_{g1}) is co-located with an asset (a_s), while an authorised agent (a_{g2}) is not present. In this case, the look-ahead of the actions performed by a_{g1} will focus only on those that allow the agent to reach the place where the asset is located, while the look-ahead of the actions performed by a_{g2} will focus only on those allowing her to exit the location where the asset is placed or to enter the areas where the asset is not located. For example, to check security requirement *SR2*, assuming the topological configuration shown in Figure 1 when the server is in *O2*, look-ahead of agents' actions will only include *Mallory* and *Alice* entering *O2*. Configurations where *Mallory* and *Alice* enter rooms different than *O2* are not taken into account. Note that this heuristic alters function f adopted in Algorithm 1, which specifies the locations on which an agent can exercise a capability, by considering only those locations containing

assets and agents that are relevant for the satisfaction of security requirements.

Another technique that can be employed is related to the threat analysis strategy. To counter the complexity explosion that can occur in certain configurations, the system might opt for a reactive mode of operation. Since the theoretical number of LTS states can be maximal in particular circumstances, the proactive approach presented previously might not be effective *at runtime*. Informally, this is likely to occur when the topological configuration requires the look-ahead of interaction of multiple agents with several parallel locations. In such a case, if a likely state explosion is observed in a part of the topology a certain change in operation can be performed to mitigate it.

To handle these situations the system has to switch to a reactive operation mode. More precisely, agents' actions are not considered and by default any action is assumed to determine a threat. Therefore, all locations are considered locked and agents have to request access in order to be able to perform any movement. This way, security requirements verification is performed when agents are attempting to move (e.g., entering co-located areas or exiting from their current place). In this case, the LTS is constructed only with respect to the requested capability execution. Our approach decides to switch to this reactive mode in case the state space exceeds a certain pre-determined threshold. However, the reactive mode is not always sustainable as it can compromise the system's usability by requiring each user to request access on every attempt to change location.

VI. TOPOLOGY AWARE PLANNING

This section illustrates the algorithm adopted to identify alternative configurations of security controls able to prevent potential violations of security requirements. A candidate configuration is selected among those that also satisfy relevant functional requirements.

A. Security Controls Identification

Algorithm 2 uses the results obtained during the threat analysis to identify alternative configurations of security controls. This recursive algorithm prunes the LTS by progressively removing states where security requirements are violated. This process essentially removes future paths of execution that should never be reachable from the current topology. R , S and S_v indicate the transition relation of the LTS, its states, and the set of states that violate the requirements, respectively. Note that each state s of the LTS that is generated during the threat analysis has exactly one predecessor. Signature $R(s)^{-1}$ returns the predecessor of state s . Recall that each LTS transition represents the execution of a capability by an agent on a specific location (e.g., $\langle Eve, in, O6 \rangle$ corresponds to transition a in Figure 3a). However, a similar transition might occur in other parts of the LTS (e.g., the one entering in state 3). We refer to such transitions that refer to the same agent, action, and location, but occur on different parts of the LTS as *homologous*. For each transition entering the violating states (S_v), all its homologous transitions (including itself)

are removed, including their successor states. For example, for transition a in Figure 3a, states 3, 4, and 6 are removed.

Algorithm 2 Identification of Security Controls

```

1: function IDSECURITYCONTROLS( $R, S_v, S$ )
2:    $sc = \{\{\}\}$ ;
3:   for  $s \in S_v$  do
4:      $S'_v = S_v \setminus s$ ;
5:      $S' = S \setminus s$ ;
6:      $s' = R^{-1}(s)$ 
7:      $c = cap(s', s)$ ;
8:      $\langle S'_v, S, R' \rangle = \text{PRUNE}(S'_v, R, S, c)$ 
9:     if  $S'_v \cap S = \emptyset$  then
10:       $sc = sc \cup \{c\}$ ;
11:    else
12:       $sc' = \text{IDSECURITYCONTROLS}(R', S'_v, S')$ ;
13:      for  $sc'' \in sc'$  do
14:         $sc = sc \cup (sc'' \cup \{c\})$ ;
15:      end for
16:    end if
17:  end for
18:  return  $sc$ ;
19: end function

```

The set sc contains the configuration of security controls that is computed in the current iteration of the IDSECURITYCONTROLS procedure. First, the algorithm creates sc (Line 2). If there are no states that violate the security requirements, the empty set is returned (Line 18), as no security controls should be applied. Each state s that violates a security requirement (Line 3) is iteratively removed from the set of violating states (Line 4) and from the set of states of the LTS (Line 5). To make this state not reachable, its predecessor s' (Line 6) is analyzed and the capability c that labels the transition from s' to s is identified (Line 7). The graph is pruned by removing all the transitions that are homologous to c (Line 8). This represents the effect of the security control associated with c , which revokes from an agent the permission to perform the action represented by c . If the new set of violating states (S'_v) does not contain any other state (Line 9), it means that forbidding the capability c (alone) allows the satisfaction of the security requirement, and thus it is added to the set of security controls to be returned (Line 10). Otherwise, additional security controls must be enforced. Thus, function IDSECURITYCONTROLS is recursively called over the new set of states S' and violating states S'_v (Line 12). When the set of security controls is returned, each element sc'' of the set of computed constraints sc' is analyzed (Line 13) and enriched with the capability c (Line 14). Note that this procedure can also terminate before all transitions entering in the dark states (S_v) are considered, since some of the violating states targeted by the remaining transitions might have already been removed due to their presence as successors to transitions already considered. The transitions used for pruning the LTS identify a configuration of security controls: the actions indicated by each transition are those for which authorisation should be revoked from the corresponding agent.

However, recall that by removing the action that leads the system to a violating state, the successors of the corresponding

homologous transitions are also removed. In other words, not all the transitions entering violating states are necessary for a configuration of security controls. Furthermore, the ordering in which transitions are considered can lead to the identification of different sets of security controls. For this reason, Algorithm 2 returns alternative sets of security controls, for each ordering in which the transitions entering in the violating states (S_v) can be considered.

- In the first scenario (Figure 3a), to satisfy security requirement $SR1$, security controls will forbid $\langle Eve, in, O6 \rangle$ or $\langle Bob, in, O6 \rangle$, depending on whether a transition to state 4 or 6 is considered first, since both of these transitions lead to violating states.
- In the second scenario (Figure 3b), $\langle Mallory, in, O2 \rangle$, will also be included in sets of security controls, to comply with security requirement $SR2$.
- In the third scenario (Figure 3c), following the identification of violating states, $\langle Trudy, in, LT1 \rangle$ and $\langle Trudy, in, LT2 \rangle$ will additionally be forbidden, to comply with security requirement $SR3$.

B. Security Controls Selection

This section illustrates possible criteria for selecting one of the alternative configurations of security controls identified by Algorithm 2. Even though different criteria can be employed, such as minimisation of the number of security controls, in this paper we propose a *requirements-driven* criterion that aims to exclude those configurations of security controls forbidding the satisfaction of non-security requirements (i.e. functional requirements). Since we assume that all the functional requirements have a fixed structure ($EF a_g \text{ in } l$), which requires the existence of a path that allows an agent a_g to reach location l , we can exploit this structure to detect the set of security controls that do not contrast with these requirements. In other words, the main idea is to compute the set of capabilities that the agent a_g must perform to access l and to filter the security controls that do not remove these capabilities.

Considering our case study and the system functional requirements (7a-7c) we notice that some configurations of security controls violate stated functional requirements. For example, the configurations that revoke from Bob the authorisation to access area O6 violate functional requirement $FR2$, which states that there should always exist a path that allows Bob to reach office O6. To guarantee the satisfaction of this requirement candidate configurations of security controls should never revoke Bob the authorisation to traverse the areas necessary to reach O6 from his location. Note that the security controls selection can be performed together with the identification of security controls. In this case, Algorithm 2 can terminate as soon a configuration of security controls that satisfies all system requirements is identified.

Since it may not always be possible to satisfy all the functional requirements at the same time, a selection criterion can aim to identify a configuration of security controls that satisfies the requirements having highest priority. Another alternative is to aim for less disruption to a specific subset

of agents (e.g., those having highest importance), as a kind of quality-of-service principle. For example, to satisfy security requirement *SR1*, potential security controls can revoke from Bob or Eve the authorisation to access the safe room. This happens because both are placed in A2, and can potentially be co-located in O6. If we assume that Bob has a higher priority than Eve, access to O6 will be only revoked to Eve even if she requires to enter O6. Another selection criterion can aim to minimise the number of security controls that are placed in a candidate configuration. This is motivated by the fact that the application of security controls could be expensive (e.g., in terms of energy consumption).

VII. EVALUATION

This section evaluates our proposed approach along two dimensions: (I) applicability of the approach and effectiveness of the adaptation procedure, and (II) classes of security requirements that can be handled in the adaptation process.

To evaluate the applicability and efficiency of the approach, we realised it by developing a prototype application. Starting from an Ambient Calculus formula that represents the current topology of the system and its requirements specified using CTL formulae, the application computes the security controls to be employed to protect the system. The implementation includes the set-theoretic procedure described by Mardare et al. [16], [15] and briefly introduced in Section V to generate the LTS, the algorithm to detect and select the security controls to be employed, and uses a third party prototype of the CTL model checking algorithm to identify violations of security requirements. The prototype is realised as a pure Python stand alone application using⁴ the MrWaffles CTL checker and NetworkX as a graph library backend.

Preliminary results demonstrate the applicability of the approach and encourage further investigation. For example, for our case study the process took under two seconds on a test machine Intel i5 (2.5GHz) having 4GB RAM. The set of the security controls that is returned by the procedure forbids the following actions: “Trudy in LT1”, “Trudy in LT2”, “Mallory in O2”, “Eve in O6”. Note that the reference implementation is not optimised with respect to computation time or space; these can be reduced rather significantly, for example by using state of the art model checking tools (e.g., NuSMV) as well as high performance data structures capable to handle large state spaces efficiently. Furthermore, this time can also be reduced by acting on the heuristics used to generate the state space or to select the security controls that can be chosen with respect to the specific case study.

The expressiveness of the proposed approach in terms of security policies that it can enforce was also evaluated. Cuppens and Cuppens [10] classify security policies in four categories: permission, prohibition, obligation and dispensation. In this paper security controls only support permission and prohibition as they grant or revoke from agents the authorisation to perform a specific action. However, our approach is also

amenable to support obligation and dispensation. In particular, obligation can be applied by forcing agents to perform actions that lead to a topological configuration that is less close - in terms of number of transitions - to a state where security requirements are violated. Dispensation is conceived as a permission to neglect a security requirement. It could be applied by tolerating some violations of some security requirements. This can be supported by not including during the selection of security controls the states in which a violation of security controls can be tolerated.

VIII. RELATED WORK

The role of topology [23] has been investigated in several domains that are not directly related to software engineering. A representation of topological network connections has also been adopted to manage large scale distributed systems. For example, Tapestry and Pastry construct a topology-aware overlay by choosing nearby nodes for inclusion in their routing tables [8]. Topology has also been extensively taken into account in the wireless sensor networks community, where it is used not only for sensor area placement [14], but also in the context of security. For example, adaptation in sensor network clustering in response to spam attacks has been proposed [12].

However, as far as we are aware, from a software engineering perspective, explicit focus on topology to support adaptive security has not been considered. In particular, Salehie et al. [21] proposed a requirements-driven approach for dynamically re-estimating the risk of harm depending on assets and context changes. Predetermined security controls are also adjusted at runtime depending on the varying risk of harm. Architecture-based self-protection (ABSP) [24] aims to detect and mitigate security threats based on an architectural representation of the software that is kept in sync with the running system. The architectural model provides information related to the impact of a security breach on the system and allows engineering security controls by applying specific architectural design patterns. However, this work does not take into account topological changes as a trigger for adaptation and is based on the assumption that security controls are predetermined. Our approach, instead, monitors changes in topology and reasons about their impact on the satisfaction of security requirements. This allows us to discover new security threats and deploy security controls that have not been previously planned for.

Existing work on dynamic access control has considered contextual information to adapt security policies. Samuel et al. [22] use contextual parameters, such as time and location to identify emergency situations for which users’ authorisation might require to be relaxed or made stricter. Similarly, the Organization Base Access Control (OrBAC) [10] allows associating security rules with conditions expressed on time, user’s location and previous behavior. These conditions provide the flexibility to enable/disable security rules dynamically. Unlike existing access control approaches, our work avoids specifying security rules in advance. It looks ahead at potential users’ actions that can be performed from the current topology and identifies undesired states where security requirements can be

⁴mrwaffles.gforge.inria.fr, networkx.lanl.gov

violated. Adequate security rules are identified dynamically by revoking from relevant agents the right to perform actions that can lead the system to undesired states.

The Ambient Calculus has been extended to represent security relevant properties. For example, ambient types [7] can be used to identify confidential ambients that cannot be opened. Boxed ambients [5] delimit the perimeter within which communication can take place, while boundary ambients [2] prevent information leakage. However, the decision to assign a specific type to an ambient in order to satisfy specific security properties is pre-determined and cannot be modified at runtime. Conversely, our approach is able to detect whether to allow or forbid the execution of agents' capabilities depending on potential threats brought by the current topology.

Several analysis techniques have been proposed in literature to verify if a system modelled as a set of Ambient Calculus formulae satisfies some security requirements. For example, Nielson et al. [11], [19] propose a technique to check whether an Ambient Calculus specification allows only some processes to be contained into other ones. This technique has also been employed for checking if the behaviors of firewalls are compliant with their security policies [18]. Braghin et al. [2], [3] define a verification technique able to identify violations of confidentiality determined when sensitive data can be moved outside a boundary ambient. Other work [4] verifies whether security policies expressed according to Bell-LaPadula model are satisfied in a program that leverages a specific network configuration. However, all the proposed model checking techniques have only been employed to verify security policies but they have not been adopted to assess security risks or to suggest possible security controls that can be applied for specific topologies of the operational environment.

IX. CONCLUSIONS

This paper has proposed an approach aimed to engineer topology aware adaptive security systems. A live model of the topology of the operational environment is used at runtime to look ahead at changes which represent future system states. Potential violations of security requirements are identified in future topological configurations reachable from the current one. A set of security controls is proactively applied to prevent the system from reaching those states where security requirements can be violated. Our approach leverages existing formalisms such as the Ambient Calculus to represent the topology and CTL to represent system requirements. Preliminary results demonstrate the viability of the approach in identifying security requirements violations and appropriate security controls for a realistic physical access control scenario. We aim to improve the expressiveness of our approach in order to manage obligation and dispensation security policies, and ameliorate the performance by building on state of the art model checkers. We believe that the approach presented contributes to opening new dimensions in reasoning on systems where topology is a first class entity. Whereas so far the focus has been on physical systems, we aim to apply this initial

approach to digital topologies to prevent threats that can arise in digital and cyber-physical systems.

REFERENCES

- [1] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. A Data-oriented Survey of Context Models. *SIGMOD Record*, 36(4):19–26, 2007.
- [2] C. Braghin, A. Cortesi, and R. Focardi. Security Boundaries in Mobile Ambients. *Computer Languages, Systems & Structures*, 28(1):101–127, 2002.
- [3] C. Braghin, A. Cortesi, and R. Focardi. Information Flow Security in Boundary Ambients. *Information and Computation*, 206(2):460–489, 2008.
- [4] C. Braghin, N. Sharygina, and K. Barone-Adesi. A Model Checking-Based Approach for Security Policy Verification of Mobile Systems. *Formal Aspects of Computing*, 23(5):627–648, 2011.
- [5] M. Bugliesi, G. Castagna, and S. Crafa. Boxed Ambients. In *Proc. of the 4th Int. Symp. of Theoretical Aspects of Computer Software*, pages 38–63, 2001.
- [6] L. Cardelli and A. D. Gordon. Mobile ambients. In *In Proceedings of POPL'98*. ACM Press, 1998.
- [7] L. Cardelli and A. D. Gordon. Types for Mobile Ambients. In *Proc. of the 26th Symp. on Principles of Programming Languages*, pages 79–92, 1999.
- [8] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks. In *Future Directions in Distributed Computing*, pages 103–107. Springer, 2003.
- [9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT press, 1999.
- [10] F. Cappens and N. Cappens-Boulahia. Modeling contextual security policies. *Int. Journal of Information Security*, 7(4):285–305, 2008.
- [11] R. R. Hansen, J. G. Jensen, F. Nielson, and H. R. Nielson. Abstract Interpretation of Mobile Ambients. In *Static Analysis*, pages 134–148. Springer, 1999.
- [12] C.-T. Hsueh, Y.-W. Li, C.-Y. Wen, and Y.-C. Ouyang. Secure adaptive topology control for wireless ad-hoc sensor networks. *Sensors*, 10(2):1251–1278, 2010.
- [13] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50, 2003.
- [14] M. Kwon and S. Fahmy. Topology-Aware Overlay Networks for Group Communication. In *Proc. of the 12th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 127–136, 2002.
- [15] R. Mardare and C. Priami. Computing the Accessibility Relation for the Ambient Calculus. 2003.
- [16] R. Mardare, C. Priami, P. Quaglia, and O. Vagin. Model checking biological systems described using ambient calculus. In *Computational Methods in Systems Biology*, pages 85–103. Springer, 2005.
- [17] R. Milner. The polyadic pi-calculus: a tutorial. Technical report, Logic and Algebra of Specification, 1991.
- [18] F. Nielson, H. R. Nielson, R. R. Hansen, and J. G. Jensen. Validating firewalls in mobile ambients. In *CONCUR99 Concurrency Theory*, pages 463–477. Springer, 1999.
- [19] H. R. Nielson and F. Nielson. Shape analysis for mobile ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 142–154. ACM, 2000.
- [20] L. Pasquale, C. Ghezzi, C. Menghi, C. Tsigkanos, and B. Nuseibeh. Topology Aware Adaptive Security. In *Proc. of the 9th Int. Symp. on Software Eng. for Adaptive and Self-Managing Systems*, 2014.
- [21] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh. Requirements-Driven Adaptive Security: Protecting Variable Assets at Runtime. In *Proc. of the 20th Int. Requirements Engineering Conf.*, pages 111–120, 2012.
- [22] A. Samuel, A. Ghafoor, and E. Bertino. Context-Aware Adaptation of Access-Control Policies. *IEEE Internet Computing*, 12(1):51–54, 2008.
- [23] J. Wang and G. M. Provan. A Comparative Analysis of Specific Spatial Network Topological Models. In *Proc. of the 1st Int. Conf. on Complex Sciences*, pages 1514–1525, 2009.
- [24] E. Yuan, S. Malek, B. R. Schmerl, D. Garlan, and J. Gennari. Architecture-Based Self-Protecting Software Systems. In *Proc. of the 9th Int. Conf. on Quality of Software Architectures*, pages 33–42, 2013.

Openness and Requirements: Opportunities and Tradeoffs in Software Ecosystems

Eric Knauss

Department of Computer Science and Engineering
Chalmers | University of Gothenburg, Sweden
eric.knauss@cse.gu.se

Daniela Damian, Alessia Knauss, Arber Borici

Department of Computer Science, University of Victoria
PO Box 1700, STN CSC, Victoria, BC V8W 2Y2, Canada
{danielad, alessiak, borici}@uvic.ca

Abstract—A growing number of software systems is characterized by continuous evolution as well as by significant interdependence with other systems (e.g. services, apps). Such software ecosystems promise increased innovation power and support for consumer oriented software services at scale, and are characterized by a certain openness of their information flows. While such openness supports project and reputation management, it also brings some challenges to Requirements Engineering (RE) within the ecosystem. We report from a mixed-method study of IBM®’s CLM® ecosystem that uses an open commercial development model. We analyzed data from from interviews within several ecosystem actors, participatory observation, and software repositories, to describe the flow of product requirements information through the ecosystem, how the open communication paradigm in software ecosystems provides opportunities for ‘just-in-time’ RE, as well as some of the challenges faced when traditional requirements engineering approaches are applied within such an ecosystem. More importantly, we discuss two tradeoffs brought about the openness in software ecosystems: i) allowing open, transparent communication while keeping intellectual property confidential within the ecosystem, and ii) having the ability to act globally on a long-term strategy while empowering product teams to act locally to answer end-users’ context specific needs in a timely manner.

Index Terms—requirements engineering; software ecosystem; mixed method

I. INTRODUCTION

Software ecosystems promise to address the inherent complexity of large-scale software projects by removing the need for centralized management [1]. By opening up and attracting 3rd-party actors into joining a software ecosystem, an organization increases its innovation power and market reach [2]. The strategies to initiate and maintain a software ecosystem vary in their degree of *openness* and range from following widely proprietary, closed information flows around a defined set of partners (e.g. SAP) to the more open information flows found in open source ecosystems (e.g. IBM’s Eclipse) [3].

More recently a prevalent strategy along this spectrum of openness in ecosystems is the open commercial approach [5] (a.k.a Extended Software Enterprises [6]), where organizations open up internal information about the product, development process and project communication, while maintaining a commercial licensing and copyright model to protect some of the organization’s intellectual property. Abolishing selected barriers the ecosystem facilitates building of communities

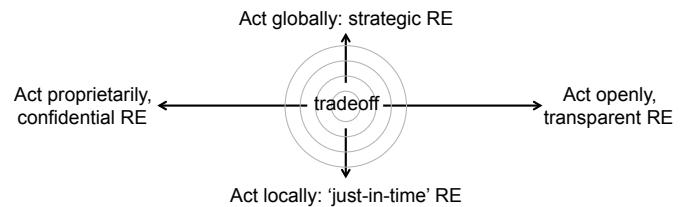


Fig. 1. Tradeoffs in open commercial software ecosystems.

and product adoption, and provides advantage over the non-open competitors [6], [7]. The increased transparency supports learning from observation and reputation management [8].

The openness in these types of software engineering environments would seem to support decentralized ways of facilitating the flow of requirements among stakeholders, similar to those found in open source projects [7]. However, open-commercial enterprises are different from open source projects. The scale and complexity of stakeholder relationships – governed by contractual considerations and commercial business models – creates problems in managing diverse sources of requirements and information flows, while the ecosystem’s openness results in lowered ability to protect product strategic information. To the best of our knowledge no systematic study examined RE in software ecosystems, although systematic literature reviews have reported on many other areas concerning software ecosystems [9], [10], leaving the description of RE processes within software ecosystems an outstanding research problem [11]. In this paper, we report from a mixed-method empirical study of IBM®’s open-commercial software ecosystem CLM® (Collaborative Lifecycle Management) to identify requirements engineering challenges and practices in an open-commercial software ecosystem.

Our contribution is two-fold: First, we provide a detailed description of RE processes and how requirements flow through the IBM CLM software ecosystem; specifically, we identify challenges around stakeholder selection, communication and prioritization of requirements, managing context, and mapping requirements to the ecosystem’s actors. Secondly, we identify two main underlying tradeoffs in open commercial ecosystems (Figure I): (1) Maintaining the openness and transparency in the ecosystem needs to be balanced with keeping the confiden-

tiality of paying customers' business needs and (2) responding to market demands requires the ability to globally define requirements on the strategic level, while scale and complexity of software ecosystems require self-organized teams' ability to locally and timely address context-specific customer needs 'just-in-time'. These insights, while developed from investigating the IBM's CLM ecosystem, have implications for other software ecosystems in which information about proprietary products, their features and development processes is being maintained in channels with some degree of openness.

II. BACKGROUND AND RESEARCH QUESTIONS

A. The IBM CLM Software Ecosystem

The IBM Collaborative Lifecycle Management (CLM) tool suite is characterized by a) the *Jazz platform*[®], a service oriented platform based on open standards, b) its open communication channels that allow all stakeholders to participate in discussions of work in progress, and c) its goal to be a strategic platform for its customers and to support seamless interaction between tools and phases in the software development lifecycle. Jansen et al. refer to such systems as software ecosystems and define them as follows [12]:

"A software ecosystem is a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artefacts."

The CLM ecosystem consists of a number of products (*set of actors in a shared market*) especially designed to interoperate on the Jazz platform (*technological platform*). These products include¹ Rational Requirements Composer (RRC), Rational Quality Manager (RQM), Rational Software Architect (RSA—now: Design Manager), Rational Team Concert (RTC), and smaller products like Reporting which integrates Rational Insight capabilities in the ecosystem. IBM Rational[®] (an IBM brand) coordinates this ecosystem and offers integration adapters to connect the CLM ecosystem to other major players and ecosystems in the field. IBM Rational also runs a partner program to allow third parties to offer products with certified compatibility to the Rational CLM products. For this reason, customers can have highly customized environments defined by the mixture of Rational and third party products that work together in order to provide the required services.

The CLM ecosystem is particularly interesting for investigating RE in software ecosystems, because of its open communication channels that allow transparent developer-customer communication: (i) a publicly available issue tracker with rich discussion forums around features, (ii) wikis with technical documentation from CLM developers, and (iii) open chat rooms. They enable customers to articulate doubts about a certain change as well as developers to learn from end-user comments about their specific needs and how they use different services in the software ecosystem. Developers also

openly discuss features they are currently developing based on a large-scale agile development process and by this they actively promote the ecosystem and its actors [5].

B. An Illustrative Example

We will illustrate the complex RE environment in the CLM ecosystem with the following (fictive but realistic) **running example**:

The CLM ecosystem offers value for lifecycle management in all kinds of engineering projects. Yet, there is a strong focus on software development. Consider Alice Inc., a new actor in this ecosystem that wants to leverage the existing services for requirements management, quality management, version control, task management, and team collaboration to build a solution for the automotive domain. Alice Inc. aims at integrating existing tools and services in that domain like specialized requirements and quality management solutions, or domain specific artifacts like Simulink models. Alice Inc. could offer CLM based services in a cloud based setup, but also based on local installation at a customer's site. Both variants offer similar challenges for the RE landscape as discussed in this paper.

C. Requirements Engineering in Software Ecosystems

We examine practices and challenges of requirements management in the CLM's open commercial paradigm based on Jansen et al.'s proposal to analyze software ecosystems on three different scope levels [6]. A full and complete description of any software ecosystem is very difficult to achieve. In Figure 2, we describe the entities and relationships in the CLM ecosystem that serve our research investigation. Section IV describes the workflows depicted in the main part of Figure 2 as well as the practices and challenges we identified on each scope level. At the top of Figure 2, the CLM ecosystem is shown on Scope Level 1 in the context of neighbouring ecosystems such as Github and SAP (circles group actors to ecosystems), on Scope Level 2 with a focus on its internal structure, and on Scope Level 3 with a focus on a single actor (here: RQM) and its direct neighbours. Links between actors indicate that the actor on the right uses services of the actor on the left.

Scope Level 1 is the highest level of abstraction and offers an external view on the ecosystem (Figure 2, top-left). The scope at this level is defined based on differentiating features such as the technological platform, the target market, the participants, and its connectedness to other software ecosystems. The figure shows CLM ecosystem actors (RTC, RRC, RSA, RQM, Reporting) as well as actors of adjacent software ecosystems around Git, Jira, or SAP platforms.

RE can have an impact on Scope Level 1 by facilitating integration or attraction of new actors into the ecosystem as well as approaching new markets, e.g. by adding strategic features to the roadmap or adjusting the technological platform. Further, these requirements need to be communicated throughout the software ecosystem.

¹<https://jazz.net/products/clm/> – visited 2014-3-4

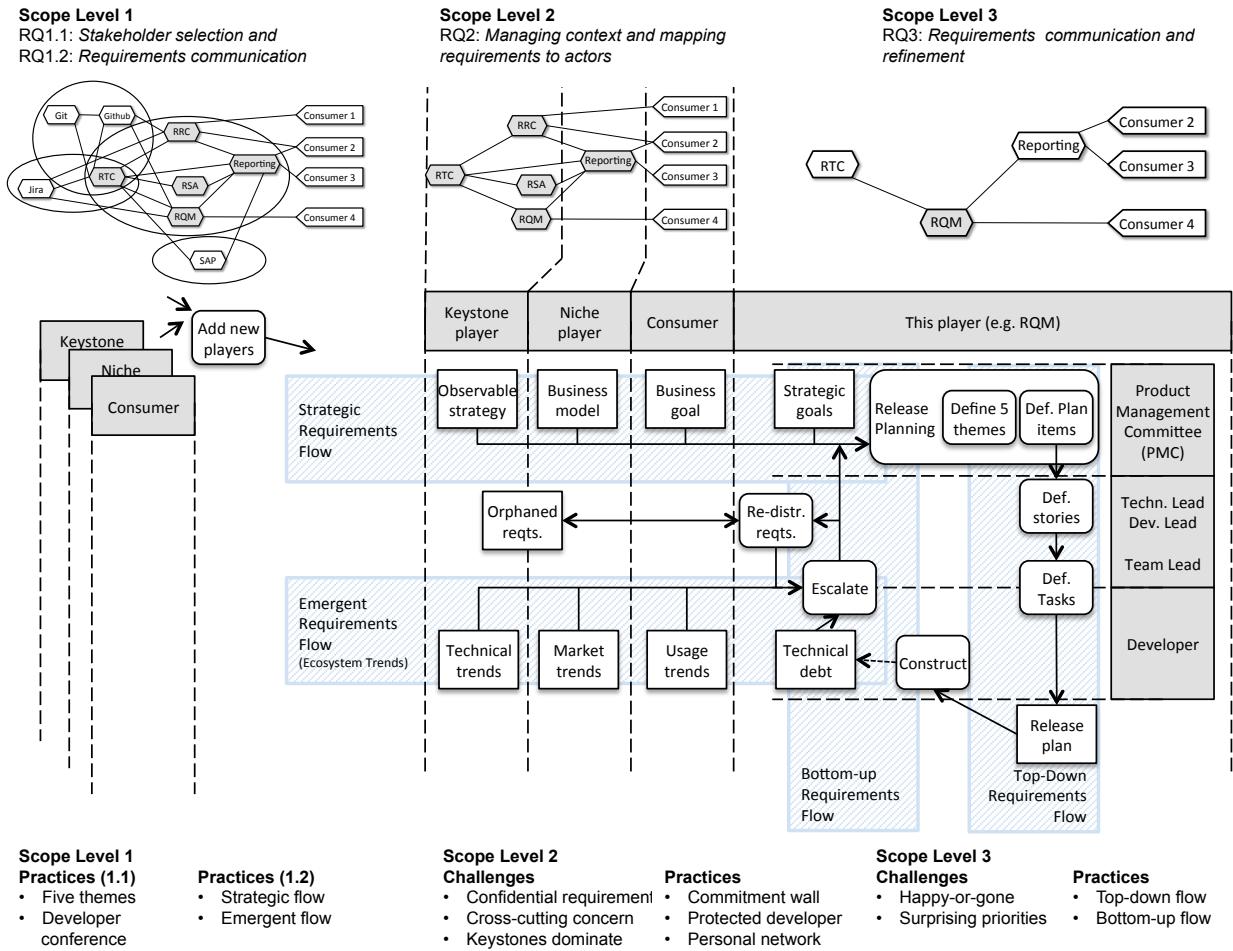


Fig. 2. Research questions, findings (flow of requirements, challenges, practices) in the context of software ecosystem scope levels.

Example: Alice Inc. might require certain features in the existing products to make CLM services attractive for the automotive domain. If the CLM ecosystem (by purpose or because of other priorities) fails to prioritize these features on its roadmap, or if the technological platform hinders Alice Inc. in offering stable services to its customers, its economic survival might be endangered. If Alice Inc. is perceived as a strategic partner, its requirements need to be communicated to all relevant actors, such as RTC or even the Jazz platform.

Previous work proposed that stakeholders, their relation, and their communication can be analyzed as requirements value chains [11] or based on social network analysis [13]. Yet, how stakeholders are selected and how requirements flow through software ecosystems in practice remains an open question:

RQ 1.1: How are stakeholders selected across the software ecosystem?

RQ 1.2: How are requirements communicated across the software ecosystem?

Scope Level 2 offers an internal view on the ecosystem (top-middle in Figure 2) that allows analyzing roles of the actors and their relations. Jansen et al. [6] distinguish three

roles of actors in software ecosystems: *Dominators* control value capture and creation of the ecosystem and tend to expand by taking over functions of other ecosystem actors. By eliminating those other actors they reduce diversity in the ecosystem and are therefore considered harmful [14]. *Keystone players* create or occupy highly connected hubs of actors. They are considered beneficial to the ecosystem health by providing value to surrounding actors as well as increasing variability and robustness [14]. *Niche players* draw value from the keystones and aim to separate from other niche players by developing special functions [14]. To support our perspective on RE, we include *consumers* as a fourth role, characterized by consuming services without offering own services in the scope of the ecosystem. Table I gives an overview of actors, their roles, and their typical stakeholders in the CLM ecosystem.

From a RE perspective, it is important to understand how actors position themselves as niche or keystone players. Related work suggests that a significant amount of functionality in software ecosystems is based on the interplay of several subsystems [15]–[17]. The way different actors offer services to consumers by re-using other services in the ecosystem de-

TABLE I
STAKEHOLDERS IN THE SCOPE OF DIFFERENT CLM ACTORS.

| Actor | Stakeholder | Role in RE activities |
|---------------------------------------|-------------|--|
| Consumer (e.g. a customer) | Manager | Source of strategic reqts. and goals |
| Keystone (e.g. RTC, RQM, RSA, RRC) | End-user | Source of specific requirements from daily usage |
| | Developer | Receiver of requirements for implementation |
| | Senior dev. | Source of requirements in relation to technical debt |
| | Techn. lead | Source of technical requirements, e.g. from architectural considerations |
| | Dev. Lead | Assigning requirements for implementation and coordination of work. |
| | Support | Facilitator of requirements flows from customers. |
| Productmgr. | | Source of strategic requirements for the product. |
| Niche (e.g. Reporting) | | Same as keystone, but highly affected by technical decisions of keystones. |

fines the context in which the consumer's requirements evolve and how difficult it will be to map those new requirements to a specific service.

Example: *A consumer from the automotive domain would probably issue feature requests to Alice Inc., who is providing CLM ecosystem services to this new domain. Those feature requests can either be handled by the new actor directly or by one of the more fundamental service providers such as RTC or RQM. It is not clear however, how they are forwarded to the actor in the ecosystem that can handle them most effectively. Also, the actor who is finally working on the feature request does not necessarily have the required domain knowledge.*

The open-commercial paradigm provides end-users with a set of different channels to articulate changed or new needs [5], but challenges end-users and customers to send such requests to the correct recipient [15], [16]. The fact that software ecosystems are constantly evolving causes the ideal recipient for a given request to change over time, thus making it necessary to rethink the way requirements travel through the software organizations and which roles are responsible to seek alignment of goals [11]. This alignment of goals requires understanding requirements in their context, for which related work focuses on contextual techniques and observation. Requirements can be captured together with their relevant context by observing end-users interacting with their system [18], [19]. Feedback systems allow end-users to articulate their needs themselves, enriched with screenshots and other context descriptions [20]. In addition, mobile devices allow to present and discuss scenarios in situ, i.e. in the context of use [21]. It remains an open question how such approaches can be applied by practitioners to manage the context of requirements and to support software managers and developers to actively align their development efforts with needs of relevant stakeholders

(e.g. by mapping feature requests to particular actors in the ecosystem) at the scale of modern software ecosystems.

RQ 2: How are requirements mapped to ecosystem actors and how is the context of requirements managed?

Scope level 3 offers an organization centric perspective. Figure 2, top right corner shows the immediate surrounding of the RQM actor. This scope level allows to investigate RE of single actors on a more tactical or operational level.

For RE in software ecosystems, it is unclear how actors can systematically understand the level of satisfaction of their external stakeholders (customers and end-users) and forward this information to internal stakeholders (software managers and developers) in a complex software ecosystem environment.

Example: *Even for Alice Inc., it is difficult to understand the satisfaction of a potentially large and heterogeneous stakeholder landscape in the automotive domain. More central actors like RTC or RQM might in addition suffer from the fact that important feedback is only given to other actors.*

Agile and continuous development, especially on a large scale, implies continuous clarification of requirements by software managers and developers throughout the development lifecycle [22]. Especially in distributed development, effective communication of requirements requires a high level of transparency [23], [24] to address the challenge of achieving a good representation of a large and heterogeneous set of stakeholders. Traditional RE methods have been reported to be insufficient to support such wide audience requirements elicitation [25]. It is not clear how a heterogeneous and potentially disagreeing audience that differs in their power to influence decisions can be integrated in the communication and refinement of requirements in open commercial ecosystems.

RQ 3: How are requirements communicated and refined in the scope of an actor in the software ecosystem?

III. RESEARCH METHODOLOGY

We employed a mixed-method research methodology to examine the characteristics of the open-commercial development in the CLM ecosystem. Our data collection methods included participatory observation, semi-structured interviews and analysis of software repositories.

1) Participatory observation: To obtain a broad view on the actors in the ecosystem and also more in-depth knowledge with the development processes and practices within one specific actor of the ecosystem, one of the co-authors worked as an intern and developer in the Reporting team at IBM Ottawa for two months. During that time, he observed work practices through his direct involvement in solving technical issues but also through participation in project meetings. He kept a daily journal and logged information on how requirements were analyzed by the development team within this ecosystem actor but also details of interactions with a number of other internal and external actors in the ecosystem. This in-depth involvement with the development team within one actor in the ecosystem was invaluable in selecting interviewees for the next phase in our data collection.

2) *Semi-structured interviews*: In order to explore the RE landscape in software ecosystems, we conducted a series of 13 semi-structured interviews. Participatory observation allowed us to get in contact with developers and managers within a number of IBM internal actors of the CLM ecosystem that had a dependency with the Reporting team. We selected interviewees on different levels, including six developers, two development leaders, four technical leaders, and one team leader. Experience of our interviewees in their role ranged from 6 month to 15 years, with an average of 2-3 years. Our interviews were based on a semi-structured interview guide² that covered our research questions, but allowed us to pursue related topics our interviewees brought up [26].

The interviews were conducted by the co-author who also did the participatory observation. The other co-authors iteratively analyzed the interview data based on the thematic analysis approach [27] as follows: We transcribed and read recordings from all interviews to get familiarized with the data. One of the co-authors then coded the data from the perspective of the research questions and started searching for initial themes by grouping the initial codes. In a series of workshops, we then reviewed the initial themes, regrouped and refined them by cross checking the interview data with the generated codes and finally established a set of themes, consisting of both challenges and practices. We defined a label for each theme and classified it based on scope level and process step. Based on this we report on the identified challenges and practices from the perspective of a software ecosystem and in the context of the underlying process.

3) *Analyzing online repository of workitems*: One of the richer source of information in this study was the ecosystem itself. Actors of the CLM ecosystem develop their products by using the CLM product suite itself and developers can be considered as end-users (“developer-end-users”) in the scope of our study. As a consequence of its open commercial development model, all of its requirements and tasks are documented as workitems in the integrated issue tracker and are available online. By querying and analyzing this repository we were able to triangulate findings from the other two methods. For example, we were able to query the issue tracker for all user stories with attachments from external partners and confirm the statement from one of the interviews that while many companies consider such data to be confidential, others share them openly. In our analysis we analyzed workitems and their meta-data (name and if available the affiliation of the owner of workitems, comments and attachments), as well as the discussion of all issues related to the workitem’s clarification, coordination, and implementation.

IV. FINDINGS

Our findings (Figure 2) include the flow of requirements and the themes we identified in our analysis of RE in the CLM ecosystem. For the ecosystem’s Scope Levels 2 and 3,

²Examples of guiding questions: How are requirements elicited and communicated to you? How do you prioritize requirements? How do you deal with context in requirements engineering for software ecosystems?

our themes include both challenges and practices. For Level 1, however, our report focuses on practices for our interviewees, software managers and developers in various ecosystem actors, although able to describe the larger context of their work, were not aware of specific challenges at this high level.

RQ1.1: How are stakeholders selected across the ecosystem?

Software product release planning has been extensively discussed (e.g. [28]). In our interviews we focused on the specifics of creating a roadmap for a software ecosystem actor and found two practices that support the selection of stakeholders: The first practice, which our interviewees refer to as *five themes*, supports roadmapping and selecting features for the next release, while the practice *developer conference* facilitates decisions about the technological platform. Since software ecosystems are inherently open to some degree, stakeholder selection is done mostly indirectly by announcing a roadmap that is attractive to certain actors, or by adjusting the technological platform to make it easier to join the ecosystem. Thus, *five themes* and *developer conference*, which we discuss in more detail here, influence the willingness and ability of players to participate in the ecosystem.

Five themes: Each IBM internal actor is managed by a Product Management Committee (PMC), consisting of technical leads, development leads, and product managers as well as representatives from support and sales. This PMC sets the goal for the next release by defining *five themes* based on strategic considerations to guide the definition of the roadmap for the next release. By including sales and support, knowledge about strategic needs of potential new actors are included in this discussion. These *five themes* can be considered as the primary steering instrument, which ultimately allows to open the ecosystem for new actors by including or excluding crucial requirements:

“Having those high level themes helps us to frame, looking at individual requirements and saying, yeah, actually that goes along with this sort of cluster of requirements.”

Open communication of the *five themes* for the next release cycle allows the alignment of several actors and to shape the attractiveness of the ecosystem to new actors.

Developer conferences: Conferences aim at bringing together technical leaders of all actors in the ecosystem to allow discussion of future directions of the underlying platform and concepts of the ecosystem, including actor-interdependencies. Thus, they play a major role in framing the software ecosystem on Scope Level 1.

“[At] innovate conference [we] get this senior technical team together regularly. We take advantage of that by spending three solid days planning for the next release.”

Technical leaders who participate in these yearly events come from all ecosystem actors (keystone players and consumers) and meet for three days. Many of them are part of the PMCs (or equivalent in non-IBM actors) and can shape the *five themes* and the roadmap accordingly.

RQ1.2: How are requirements communicated across the software ecosystem?

We found two general information flows that introduce requirements to the ecosystem. The *strategic requirements flow* takes into account business goals and global strategies, while the *emergent requirements flow* results from local just-in-time RE activities and the open communication paradigm (Fig. 2).

Strategic requirements flow: Business goals from consumers as well as strategic information from other actors in the ecosystem are introduced into the ecosystem via sales or support activities and need to be considered during release planning. As in traditional RE (e.g. [28], [29]), this flow focuses on systematic analysis of business goals from stakeholders and their refinement to detailed requirements. The ecosystem adds an increased need to take into account strategies of other keystone players and to anticipate their movement in the ecosystem. Business models of niche players can be considered, e.g. to understand how those can contribute to the own strategy. For example, the niche player Reporting offers additional business value for many CLM products.

Emergent requirements flow: The complexity of the ecosystem causes many new requirements to be based on *ecosystem trends* and introduced into the ecosystem as *emergent requirements flow*. Sources are end-users at consumers, who find new ways of using existing services and developers at other keystone or niche players, who come up with innovative solutions. Such trends become visible in discussions of low-level workitems in open communication channels and need to be fitted into the different actors' plans, e.g. by "squeezing them into the scope of one plan item" as one team lead put it. According to our interviews, roughly $\frac{2}{3}$ of the end-user requirements originate from end-users at the customer sites, while $\frac{1}{3}$ originate from CLM developer-end-users. In addition to this, technical debt and bugs emerge as the ecosystem evolves and are communicated partly internally and partly on open communication channels. This requirements flow resembles the requirements practice in open source projects [30] where requirements emerge as informalisms, through continually emerging webs of software discourse (e.g. email and discussion forums).

RQ2: How are requirements mapped to actors in the software ecosystem and how is the context of requirements managed?

Our findings for this research question are shown in the center of Figure 2 (Scope Level 2). We found managing context and mapping requirements to actors to be two highly interconnected and challenging RE tasks on Scope Level 2 and identified three challenges that express how constraints and complexity of open-commercial software ecosystems affect the engineering of requirements:

Confidential requirement: In a commercial setting, customer specifics and requirements often cannot be discussed on open-commercial channels because they contain confidential information about the customer. Development or team leads manage this confidential information and forward it to developers as needed.

"We can work on sanitizing the requirement [and] always translate these into public workitems on jazz.net [so that] either it does not have any customer identifiable information in it or [...] the customer is okay with this."

Consequently, important information (especially related to the context) has to be removed from open communication channels, threatening the transparency of the software ecosystem.

"Informally the context is captured with one requirement, we understand this requirement is coming from organizations with this kind of environment and this kind of expectations [...] shouldn't we be capturing that context in a way that is more structured?"

Cross-cutting concern: Practitioners frequently struggle to understand if requests of different customers could be addressed generally (closer to the platform) or only specifically (by a peripheral actor). This includes dealing with cross-cutting concerns or requirements that should entirely be assigned to other actors in the software ecosystem. Especially, when several actors need to collaborate to work on such a request, a systematic approach is desirable, yet missing.

Keystone dominates: Niche players couple their value creation in a narrow niche closely to integration of other actors' services. Thus, they become dependent on these other actor's technical decisions which can have more impact on the niche player's requirements than customers or end-users.

"Most of the stuff, we just have to do it to keep up with the ecosystem and platform."

This challenge is caused by the nature of software ecosystems and significantly impacts the way requirements are mapped to niche players. Yet, niche players have little power to deal with this challenge themselves and if it is encountered too frequently, this challenge can have a serious effect on the ecosystem's overall health [14]. Keystone players should be considerate in their action. While each major design decision can offer some of the actors in the ecosystem new opportunities, it is potentially harmful to others. Keystone players need to provide some stability to niche players, because their limited resources might not allow them to follow new technical trends [14]. Yet we found no evidence for a systematic approach to monitor the requirements and needs of niche players to support decision making on this level.

Practitioners address these challenges based on three practices:

Commitment wall: As a consequence of the *confidential requirement* challenge, the release planning is partly intransparent, because confidential information like customer priorities cannot be disclosed. To mitigate potential conflicts (e.g. when a paying customer is over-bidden by another), developers are not allowed to promise a feature or bug-fix for a specific milestone or time-frame.

"We try to ensure that there is a bit of a wall in place between development and customers when it comes to commitment for enhancements [...] or for new features."

Protected developer: Open commercial channels enable developers to receive feedback from end-users directly, but

developers are widely relying on their senior team members to give them all required information. The less seniority our interviewees had, the less awareness for the special challenges of ecosystems they showed. Interception and resolution of these challenges by seniors allows developers to focus on development tasks.

“There are customer calls [taken by team or technical lead], then priorities are passed down in weekly meetings.”

In this way, team leads and other managers are effectively shielding their developers from any challenges arising from context specific requirements.

Personal network: Mapping of emergent requirements to actors in the ecosystem becomes difficult, especially because they are often cross-cutting over several players in the ecosystem. It is important that the context is well understood and that requirements are systematically forwarded to relevant other players in the ecosystem. Our interviews indicate that at the moment, this task depends on individual excellence and ad hoc coordination between seniors of several ecosystem actors.

A common practice of internal stakeholders is to make use of their *personal network*. They meet senior staff of other actors in the ecosystem and try to follow relevant developments in the ecosystem, thus being able to discover and resolve cross-cutting concerns, map (and forward) requirements to different subsystems, and to understand their general context.

“A lot of it is basically talking to the senior people on the [different] team[s].”

Some even actively track open communication channels of other actors to identify cross-cutting problems without this task being formally assigned to them.

“You need to know all workitems of the last years and their history.”

The ability of these senior team members to function in this job depends on a large part on their personal experience and network and we found no systematic approach for capturing and managing context. According to our interviewees, without such an approach, the success of an actor in the ecosystem depends too much on individuals, endangering long-term health and reliability of large-scale software development.

RQ3: How are requirements communicated and refined in the scope of an actor in the software ecosystem?

On Scope Level 3 (right hand side in Figure 2), developing teams struggle with the complexity of the software ecosystem and with the consequences of practices on Scope Level 1 and 2. For example, *protecting developers* (practice on Scope Level 2) can cause those developers to be surprised by developments in the ecosystem (see *surprising priorities* below) or cause testers to struggle with setting up a representative testframe or reproducing a bug. Another challenge on this scope level, *happy-or-gone*, refers to a lack of immediate contact with end-users and customers.

Surprising priorities: Especially junior developers heavily rely on their managers and senior developers in the team to navigate the software ecosystem challenges (see also the

protected developer practice). Consequently, it is hard for them to anticipate changes of requirements or priorities.

“You’d ask [name of technical lead] or one of the persons in charge, then you get pointed to the guys who know about an issue. [...] I would like] a head start about something becoming a priority, before it becomes that urgent thing that needs to be fixed right away.”

Happy-or-gone: Priorities become even more surprising by the huge amount of different information channels, which are only partly open, and due to the lack of systematic feedback about stakeholder satisfaction.

Strong sales and support processes are in place to understand stakeholder needs and user experience teams can provide further input, but there is a lack of transparency and systematic approaches for channeling stakeholder feedback. If there is a problem, development teams usually receive consumers’ complaints, but if there is no complaint, they do not know if a consumer is happy with a service or stopped using it.

“[Sometimes I wonder if it is] really going well for everybody or is nobody using it. And those two situations are sometimes hard to distinguish.”

In the presence of these challenges, we found practitioners relying on two general flows of requirements:

Top-down requirements flow: Members of the PMC, team leads, and developers work with specific artifacts in the CLM issue tracker, as indicated by the *top-down requirements flow* on the right hand side of Figure 2. The PMC works with so called plan items to create a release plan. Plan items are special workitems or tasks in an issue tracker. They refer to major development efforts and comprise several [user] stories (another specific workitem type). User stories are derived from plan items and assigned to team leads who then define *tasks* based on these stories and assign them to developers. During this final refinement, most of the cross-cutting concerns and context specifics are resolved, so that developers can focus on the task with minimal distraction.

“[...] PMC lead would assign me some of the plan items which I will own and which I can refine then and assign to [team lead name] or whoever else is on my team. Then refine those into stories, enhancements, and tasks that we can give to the developers to actually work on.”

The top-down requirements flow addresses traditional requirements refinement in an effective way, but is not sufficient to address the software ecosystem challenges.

Bottom-up requirements flow: In order to deal with challenges discussed on both Scope Level 2 and 3, team leads and senior developers use their personal contacts to facilitate the flow of requirements between their team and support or other development teams. Open-commercial instruments allow customers and end-users (including developer-end-users) to introduce requirements themselves or to discuss their needs in the comment stream of a workitem.

“[Customers are] either submitting workitems directly, like defects, or there is some forum / blogs, where people ask questions.”

This information is often available on a very specific and low abstraction level, and senior developers who have a good overview of such issues are consulting the PMC directly or indirectly about trends as well as technical debt.

“When we come up with these high level themes, we often come, we always come up with this grab bag called technical debt.”

In the absence of formal processes or other systematic approaches, *personal network* and experience of team leads and senior developers are the only way to understand the level of stakeholder satisfaction and to highlight technical debt that should be addressed.

“[You need to] Get a feeling about a large number of reported issues. It really depends on your experience.” —

“One thing I did was internally, just informally, poll the people who I know to be the technical leaders on the team.”

We describe this ad hoc treatment of cross-cutting concerns as *bottom-up requirements flow* (on the right in Figure 2).

From a knowledge management perspective, this is a dangerous situation, as experienced and well-connected team leads and senior developers are hard to replace. For new developers and managers the complex environment results in a steep learning curve, before they can assess stakeholder satisfaction or manage requirements flows. Managers among our interviewees agree that a more systematic solution for managing complexity of software ecosystems is one of the future challenges.

V. DISCUSSION: RE-RELATED TRADEOFFS IN OPEN COMMERCIAL SOFTWARE ECOSYSTEMS

The observations and findings in our study are of course specific to the CLM ecosystem. However, they reveal underlying tradeoffs that openness brings to how requirements are managed within software ecosystems. We discuss, for each tradeoff, the different forces that need to be balanced and ways in which IBM teams approached these challenges. We therefore intend that these tradeoffs and strategies have implications for RE practices in software ecosystems in general.

Tradeoff 1: Act Openly vs. Act Proprietary

The open-commercial approach in the CLM ecosystem means that customers and end-users, in fact every stakeholder interested in the development, has access to the issue tracker and can see the current progress of the project. End-users can comment and submit bugs and they can see the status of their requests. Yet, business needs make it necessary to treat certain information confidentially. The following *forces* need to be balanced with respect to this tradeoff.

Information flow: The open commercial approach with its open communication channels facilitates information flows between end-users, customers, developers and software managers. This high level of transparency is, one could argue, an unprecedented opportunity in how software projects are engineered and an asset in dealing with the complexity of such ecosystems.

Confidentiality: If, for example, a new report has to be created for a given customer, this will appear as a workitem in the issue tracker on jazz.net. For the development, it is important to know the context of this report: how is it embedded in the customer’s context and what exactly are the customers information needs. An example of an old report this customer is using would be helpful. However, most customers are reluctant to share such intimate information about their central business processes in an openly accessible issue tracker. For this reason, it is not possible to have all information at one place and the openness is broken.

Priorities: A special case of confidential information are priorities and their rationales, which cannot be openly shared in many commercial settings, leading to incomplete information and intransparent decisions on open channels.

General solution strategy: *Introduce layers between customers and developers.* In our case study, actors acquire sensitive, confidential context specific information through sales and support groups and share them directly with the developers in charge. They ensure that discussion in open communication channels is on a high level of abstraction, e.g. by introducing acronyms such as LUGA (Large Unknown Government Agencies) to refer to anonymous entities. We found, however, that this strategy hinders the information flow, significantly adds to management effort, and increases the challenge of creating a holistic product strategy that is in line with context specific requirements of specific customers, which we will discuss next.

Tradeoff 2: Act Globally vs. Act Locally

The CLM ecosystem is in direct or indirect competition with other lifecycle management solutions, e.g. *Visual Studio Application Lifecycle Management* [31] or *SAP Solution Manager* [32]. In order to position CLM against these competitors, strategic decisions need to be made that affect the whole ecosystem. To define a global strategy in a software ecosystem of this complexity a strict plan-driven top down approach is suitable. At the other end of the spectrum, very local decisions need to be made to adjust services to meet the ever changing consumer needs or to address technical debt. In a constantly evolving software ecosystem, these fast and agile decisions ask for local empowered development teams. The following *forces* need to be balanced with respect to this tradeoff:

Understanding customers in context: User stories and tasks can be difficult to understand, when the context is not clear [21]. Context in the CLM ecosystem depends on various factors, and descriptions of workitems frequently showed gaps due to confidential information. Consider for example a given customer with a unique setup, consisting of server and client platforms as well as a specific mixture of CLM and 3rd party products. If such a customer has a new requirement, it is often not obvious whether this requirement is only valid for this single customer or if this is a request for a feature with general value. Developers in the CLM ecosystem had clear difficulties making decisions about the customer’s specific context especially when the customer’s development process

and culture was different from those that CLM developers experience everyday, thus not being able to rely on their own domain knowledge. This situation is similar to the one found in Fricker's requirements value-chain vision paper [11], where locally empowered development teams actively pulled in requirements to offer value in the ecosystem based on their domain-expertise and context related knowledge. In fact it is unclear, if any other approach can scale for tomorrow's ultra-large scale software systems [1].

Learning curve and dependence on experience: Product and team leadership remove requirement conflicts and inconsistencies that result from the large set of stakeholders and their different contexts. The complexity of this task imposes high requirements on experience and a strong personal network across the CLM ecosystem, to allow basing decisions on the views of senior people in related projects. Hiring new software managers or promoting developers is made difficult by this steep learning curve.

Cross-cutting concerns: Empowering local teams typically increases the need to deal with cross-cutting concerns and hidden technical dependencies [33]. In the CLM ecosystem, even if discovered, such cross-cutting concerns are difficult to tackle. Descriptions of workitems are often unclear and it is difficult to identify the person that might help with clarifying. Because differences in timezones do not allow regular synchronous communication, emails or workitem comments are used to identify a developer who might provide clarification, often leading to long email threads over several days without any valuable information.

General solution strategy: *Rely on personal excellence.* We found that the challenges caused by the tradeoff between acting globally to define a strategy for the ecosystem top down and acting locally to understand and react to context specific requirements are mitigated by excellent lower and middle management. Successful managers have several years of experience of working in the CLM ecosystem and in addition a strong network throughout the keystones of the ecosystem. By this, they are able to resolve cross-cutting concerns and reassign misclassified requirements. Nevertheless, this situation is far from satisfactory. It is increasingly difficult to hire qualified managers that can be effective in a reasonable amount of time. Dealing effectively with context-specific requirements depends partly on luck and requires that the right person becomes aware of an issue at the right time. A more systematic approach to manage and distribute the contextual knowledge is needed to ensure continuous excellence. Such an approach could systematically facilitate bottom-up information flows to support global decisions in the ecosystem, and horizontal information flows to handle cross-cutting concerns between actors or even teams [33].

Threats to Validity

To mitigate threats to *construct validity* [26], we examined the CLM ecosystem using terms defined in the extensive treatment of ecosystems edited by Jansen et al. [4], and

triangulated our findings with insights from multiple methods: participatory observation, interviews and repository analysis.

With respect to *external validity*, our study has been of a particular domain of action and our findings should be regarded as tendencies rather than predictions [34]. We are confident that our results will prove useful for other similar organizations and contexts, i.e. software ecosystems that are neither fully open nor fully closed with respect to requirements related communication.

To increase the *internal validity*, our author team closely collaborated during the participatory observation phase to create the interview guide for the semi-structured interviews. All interviews were performed by the same author, who did not participate in transcription and coding of interviews, but was available for clarification questions. The other authors searched, reviewed, and defined the themes iteratively and discussed intermediate results regularly with practitioners.

VI. RELATED WORK

In understanding ecosystems, one would draw on three fields in software engineering: open source software [30], modelling and architecture (e.g. software evolution, architecture, and product lines [35]), and managerial perspectives (e.g. business aspects and co-innovation [6]). Software ecosystems imply some degree of openness and different strategies exist from widely proprietary ecosystems based on a semi-open partnership program to pure open source ecosystems [3], [7]. Related works [35]–[37] discuss how to analyze software ecosystems and relationships among their actors. While influenced by these studies, we focus on understanding implications these relationships have on RE practice and vice versa.

Literature discusses almost as many proprietary ecosystems as free-and-open-source ecosystems [10]. RE practice in traditional proprietary software projects (as e.g. described in [28], [29]) differs significantly from the way requirements are handled in open source projects, where requirements are post-hoc assertions of functional capabilities and included into the system feature set after their implementation [30]. Our study indicates that although the open commercial approach of the CLM ecosystem does include a more open way of communicating requirements than in traditional approaches to RE, the requirements processes and flows are different than in open source projects. The emergent requirements flows generated by the technical implementation at the operational level are complemented by strategic requirements flows that allow the ecosystem to consider the refinement of requirements from high-level, business goals into strategic release planning.

Transparency has been proposed as a non-functional requirement in order to address the increasing demand of society to understand digital infrastructure in the information age [38]. Our research speaks to the value of transparent requirements information in complex, evolving, commercial systems but also highlights limitations and challenges for such openness in proprietary environments. Despite these challenges, open communication channels have shown their value for building communities around healthy ecosystems [39]. For commercial

software ecosystems this offers an exciting opportunity to improve scalability by facilitating decentralized 'just-in-time' RE and to support agile development [40].

VII. CONCLUSION

In this paper we described RE challenges and practices within the CLM software ecosystem, and identified two trade-offs that openness brings about in software ecosystems. Open information channels support both global strategic and local just-in-time action, but the openness conflicts with the need to act as a reliable business partner and according to non-disclosure agreements about intellectual property and confidential information. Both global and local action is needed to make the software ecosystem a competitive business partner, but to allow for both, bottom-up and horizontal information flows need to be dealt with systematically. Our work addresses a particular lack of RE research in the rapidly growing field of software ecosystems. A good starting point for future work is the development of methods and tool support for commercial organizations to optimize their RE to address the following challenges:

- Manage stakeholder interaction across multiple organizational boundaries and between teams.
- Manage domain and technical knowledge during continuous deployment across all organizational levels/actors.
- Systematically transform requirements flows into strategic and technological decisions to position actors in the software ecosystem.

ACKNOWLEDGEMENT

We thank the various participating IBM teams, and all researchers and practitioners for their feedback on this work: the SEGAL group at University of Victoria, IBM research, and the Division of Software Engineering in Gothenburg. This research was partly funded by the NECSIS Network, Canada.

REFERENCES

- [1] P. Feiler, R. P. Gabriel et al., *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Software Engineering Institute, 2006.
- [2] A. Gawer, *Platforms, Markets, and Innovation*. Edward Elgar, 2009.
- [3] J. van Angeren, J. Kabbedijk et al. *Managing software ecosystems through partnering*, in [4], 85–102.
- [4] S. Jansen, M. A. Cusumano, and S. Brinkkemper, Eds., *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Cheltenham, UK: Edward Elgar, 2012.
- [5] R. Frost, "Jazz and the eclipse way of collaboration," *IEEE Software*, vol. 24, no. 6, 114–117, 2007.
- [6] S. Jansen, S. Brinkkemper, and A. Finkelstein, *Business Network Management as a Survival Strategy*, in [4], 29–42.
- [7] S. Jansen, S. Brinkkemper et al. "Shades of gray: Opening up a software producing organization with the open software enterprise model," *The Journal of Systems and Software*, vol. 85, p. 1495–1510, 2012.
- [8] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Leveraging transparency," *IEEE Software*, vol. 30, no. 1, 37–43, 2013.
- [9] O. Barbosa, R. Pereira et al. *A Systematic Mapping Study on Software Ecosystems through a Three-dimensional Perspective*, in [4], 87–129.
- [10] K. Manikas and K. M. Hansen, "Software ecosystems: A systematic literature review," *Systems and Software*, vol. 86, 1294–1306, 2013.
- [11] S. Fricker, "Requirements Value Chains: Stakeholder Management and Requirements Engineering in Software Ecosystems," in *Proc. of Requir. Eng.: Foundation for Softw. Quality*, Essen, Germany, 2010, 60–66.
- [12] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in *Proc. of Int'l Conf. on Softw. Eng.*, 2009, NIER Track.
- [13] D. Damian, S. Marczak, and I. Kwan, "Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks," in *Proc. of Int'l Requir. Eng. Conf.*, 2007.
- [14] K. Manikas and K. M. Hansen, "Reviewing the Health of Software Ecosystems - A Conceptual Framework Proposal," in *Proc. of Int'l Wksp on Softw. Ecosys.*, Potsdam, Germany, 2013, 33–44.
- [15] C. Castro-Herrera, C. Duan, et al. "Using Data Mining and Recommender Systems to Facilitate Large-Scale, Open, and Inclusive Requirements Elicitation Processes," in *Proc. of Int'l Requir. Eng. Conf.*, Barcelona, Spain, 2008, 165–168.
- [16] K. Schneider, "Focusing Spontaneous Feedback to Support System Evolution," in *Proc. of Int'l Requir. Eng. Conf.*, Trento, 2011, 165–174.
- [17] K. Schneider, S. Meyer, et al. "Feedback in Context: Supporting the Evolution of IT-Ecosystems," in *PROFES*, 2010, 191–205.
- [18] W. Maalej and A. K. Thurimella, "Towards a Research Agenda for Recommendation Systems in Requirements Engineering," in *Proc. of Int'l Wksp on Managing Requir. Know.*, 2009, 32–39.
- [19] O. Brill and E. Knauss, "Structured and Unobtrusive Observation of Anonymous Users and their Context for Requirements Elicitation," in *Proc. of Int'l Requir. Eng. Conf.*, Trento, Italy, 2011, 175–184.
- [20] O. Liskin, C. Herrmann, et al. "Supporting acceptance testing in distributed software projects with integrated feedback systems: Experiences and requirements," in *Proc. of Int'l Conf. on Global Softw. Eng.*, 2012.
- [21] N. Seyff, N. Maiden, et al. "Exploring how to use scenarios to discover requirements," *Requir. Eng.*, vol. 14, no. 2, 91–111, 2009.
- [22] E. Knauss, D. Damian, et al. "Detecting and Classifying Patterns of Requirements Clarifications," in *Proc. of Int'l Requir. Eng. Conf.*, Chicago, USA, 2012, 251–260.
- [23] D. Damian and D. Zowghi, "RE challenges in multi-site software development organisations," *Requir. Eng.*, vol. 8, 149–160, 2003.
- [24] D. Damian, "Stakeholders in global requirements engineering: Lessons learned from practice," *IEEE Software*, vol. 24, 21–27, 2007.
- [25] T. Tuunanan and M. Rossi, "Engineering a Method for Wide Audience Requirements Elicitation and Integrating It to Software Development," in *Proc. of Hawaii Int'l Conf. on System Sciences*, vol. 7, Jan. 2004.
- [26] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Emp. SE*, (14) 131–154, 2009.
- [27] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, 86–94, 2006.
- [28] G. Ruhe, *Product Release Planning: Methods, Tools and Applications*. CRC Press, 2010.
- [29] S. Robertson and J. Robertson, *Mastering the Requirements Process*. Addison-Wesley, 1999.
- [30] W. Scacchi, "Understanding requirements for open source software," in *Proc. of Design Reqts. Wksp.* Springer LNBIP 14, 2009, p. 467–494.
- [31] J. Rossberg and M. Olausson, *Pro Application Lifecycle Management with Visual Studio 2012*, 2nd ed. Apress, 2012.
- [32] M. O. Schäfer and M. Melich, *SAP Solution Manager*. SAP Press, 2011.
- [33] N. Sekitoleko, F. Ebvota, et al. "Technical Dependency Challenges in Large-Scale Agile Software Development," in *Proc. of Int'l Conf. on Agile Softw. Dev.*, Rome, Italy, 2014.
- [34] G. Walsham, "Interpretive case studies in is research: nature and method," *Eur. J. Inf. Syst.*, vol. 4, 74–81, 1995.
- [35] J. Bosch, "From Software Product Lines to Software Ecosystems," in *Proc. of Int'l Conf. on Softw. Product Lines*, 2009.
- [36] S. Jansen and M. A. Cusumano, *Defining Software Ecosystems: A Survey of Software Platforms and Business Network Governance*, in [4], 13–28.
- [37] K. Manikas and K. M. Hansen, "Characterizing the danish telemedicine ecosystem: Making sense of actor relationships," in *Proc. of MEDES'13*, Neumünster Abbey, Luxembourg, 2013, 211–218.
- [38] J. C. S. d. P. Leite and C. Cappelli, "Software transparency," *Business & Information Systems Engineering*, vol. 2, no. 3, 127–139, 2010.
- [39] T. Kilamo, I. Hammouda, et al. *Open source ecosystems: a tale of two cases*, in [4], 276–306.
- [40] M. Lee, "Just-in-time requirements analysis—the engine that drives the planning game," in *Proc. of Int'l Conf. Extreme Programming and Agile Proc. in Software Eng.*, Alghero, Italy, 2002, 138–141.

RISDM: A Requirements Inspection Systems Design Methodology

Perspective-Based Design of the Pragmatic Quality Model and Question Set to SRS

Shinobu Saito, Mutsuki Takeuchi

R&D Headquarters
NTT DATA Corporation
Tokyo, Japan
{saitousnb,
takeuchimtk}@nttdata.co.jp

Setsuo Yamada
Software Innovation Center
NTT Corporation
Tokyo, Japan
yamada.setsuo@lab.ntt.co.jp

Mikio Aoyama
Dep. of Software Engineering
Nanzan University
Seto, Japan
mikio.aoyama@nifty.com

Abstract— The quality of the SRS (Software Requirements Specification) is the key to the success of software development. The inspection for the verification and validation of SRS are widely practiced, however, the techniques of inspection are rather ad hoc, and largely depend on the knowledge and skill of the people. This article proposes RISDM (Requirements Inspection Systems Design Methodology) to design the RIS (Requirements Inspection System) to be conducted by a third-party inspection team. The RISDM includes a meta-model and design process of RIS, PQM (Pragmatic Quality Model) of SRS, and a technique to generate inspection question set based on the PQM and PBR (Perspective-Based Reading). We have been applying the RIS designed by the proposed RISDM to more than 140 projects of a wide variety of software systems in NTT DATA for five years. By analyzing the statistics from the experience, we discovered some key quality characteristics of SRS reveal strong correlation to the project cost and level of quality to be used for evaluating the maturity of the SRS and predicting the risk.

Index Terms— Requirements Inspection, Requirements Verification and Validation, SRS, Pragmatic Quality Model, Question Set, Risk Prediction.

I. INTRODUCTION

To assure the quality of the SRS, the inspection and review¹ are the most effective and yet common practice in the V&V (Verification and Validation) process [2, 4, 21]. However, a recent survey revealed the practices of the requirements inspection are still rather ad hoc, although many projects and organizations defined the rules and procedures for the requirements inspection [12]. The survey also pointed out that lack of the methodology for systematic design of inspection method is one of the root causes of the poor practice of the requirements inspection. Therefore, the performance of the requirements inspection largely depends on the knowledge and skill of the people.

Many projects develop their own techniques apart from the organizational standard procedure. These poor practices lead to

poor quality assurance of the requirements specifications, and bottles up the improvement of inspection techniques.

It is commonly practiced to divide of work in software development to maximize the ability of professionals. Therefore, in many projects we involved, the requirements specifications are developed by separate teams or organizations from the development teams and organizations. In such cases, it is critical to assure the quality of the SRS so that the development teams/organizations can understand the correct requirements and implement them as a software system.

The authors developed a third party inspection method and applied to real projects for several years [26]. However, the inspection method is specific to our organization and development context, and hard to apply to other organizations or different context. Therefore, it is necessary to develop a design methodology of requirements inspection method to be widely used.

This article proposes RISDM (Requirements Inspection System Design Methodology), pronounced like rhythm, a methodology to design RIS (Requirements Inspection System) to be conducted by a third party inspection team. Here, we call RIS as a system of conducting inspections to SRS with well-defined procedure and a set of coordinated techniques. Therefore, the proposed design methodology can create a RIS to SRSEs for a project.

The major contribution of the proposed RISDM is to enable a systematic design of a RIS by integrating the following techniques;

- A meta-model of the RIS and generic process for designing a RIS based on the meta-model,
- PQM (Pragmatic Quality Model): A quality model of requirements specification from a specified viewpoint of a requirements reader,
- A technique to generate the question set for inspecting an SRS based on PBR (Perspective-Based Reading),
- A set of key quality factors, identified from the five years practice of the proposed methodology, which are useful to improve the SRS and predict the risk in terms of the cost-overrun of the project.

¹ We use inspection due to the formality and rigorousness [16].

The rest of this article is structured as follows.

Chapter II discusses related works. Based on the research context and approach illustrated in Chapter III, the RISDM (RIS Design Methodology) is proposed in Chapter IV. Chapter V reports the current status of the practice of the RIS designed by the proposed methodology. Chapter VI discusses the experiences of the practice of the RIS, and reveals some key quality characteristics to improve the quality of the SRS and evaluate the risks in terms of cost overrun of the project. Chapter VII discusses the contributions of this work, followed by the conclusions and future works in Chapter VIII.

II. RELATED WORK

A. Standard and Quality Model of SRS

IEEE 830 is a widely recognized standard of SRS [15]. It also defines eight quality characteristics of an SRS. However, the quality characteristics are rather generic by its definition, and hard to apply in practical use. Furthermore, the contents of an SRS are diverse; ranging from business requirements to software requirements. In practice, it is necessary to provide a concrete definition of quality characteristics of an SRS by considering its scope and context.

Davis et al. proposed 24 attributes of software requirements specifications [6]. For more concrete quality model, Krogstie et al. defined “pragmatic quality” as a quality specific to a certain reader, say, user or developer to the SRS [20]. Fabbrini et al. extended the pragmatic quality model, and defined four layers of quality types of syntactic, structural, semantic, and pragmatic for the SRS written by natural language [9]. The pragmatic quality model is promising to evaluate the quality of SRS due to the diversity of the readers. However, conventional works do not address the techniques for the design and use of the pragmatic quality model [25]. It is necessary to develop a practical method to evaluate the pragmatic quality of the SRS.

B. Inspection Techniques of SRS

Inspection is well-defined discipline, and has been widely applied in the implementation process of software development [10]. In requirements engineering, there are a bunch of literatures on the inspection and review [1, 2, 4, 13, 16, 17, 18, 24, 27, 28]. For example, Pohl introduces four classifications of techniques of validating SRS [23]. However, the inspection and review of SRS need special attention [18]. Syntactically, it has to deal with diverse descriptions in natural language text, figures and tables. Semantically, the contents of the SRS are diverse, and need rich knowledge of the domain and context of the objective system ranging from business to software. Sommerville and Sawyer suggest to organize formal requirements inspection [30]. However, inspection of SRS in practice is time consuming and error prone, largely depends on human knowledge and reading skill [13].

On the other hand, techniques of SRS inspection are less mature than source code inspection due to the diversity of SRS [28]. To cope with the diversity, PBR (Perspective-Based Reading) is a promising technique for SRS inspection [29]. It generates question set to a document from a specific perspective, say, user, or developer. Inspectors can read the

document with the question set. However, how to design the question set is left to practitioners. The authors also developed a third-party inspection technique based on the PBR and pragmatic quality model, and practiced it to a number of real projects [26]. However, the proposed technique weak in the systematic design of question set, and requirements inspection system.

C. Impact of SRS Inspection and Quality

It is widely acknowledged that the quality of SRS is the key to success or failure of the subsequent development [7, 19, 21]. However, the evidence of the impact of quality of the SRS to the subsequent development is still scarce.

One of the early work is the analysis of statistical of NASA. It revealed a negative correlation between percent project cost for RE and percent project cost overrun [11]. Damian et al. empirically studied the complex impact of RE to other processes with the statistics collected from the commercial software development [5]. Recently, the authors revealed ROI of RE, that is, the ratio of reduction of total development cost over the requirements inspection cost [19, 25]. As a related work, measuring SRS in text is also important, but is a difficult issue [13]. Another approach is the five levels of taxonomy of the impact of RE process change [14].

All the related works touch some aspects of the quality and impact of the SRS to the subsequent development. However, we need more comprehensive analysis of the quality of the SRS and its impact to the entire development. Furthermore, practitioners require useful measure such as how the quality of SRS impacts to the cost, quality and delivery of the system, which is largely unknown in the literatures.

III. RESEARCH CONTEXT AND APPROACH

A. Research Context: A Third-Party Inspection Process

Figure 1 illustrates the key ideas of the proposed design

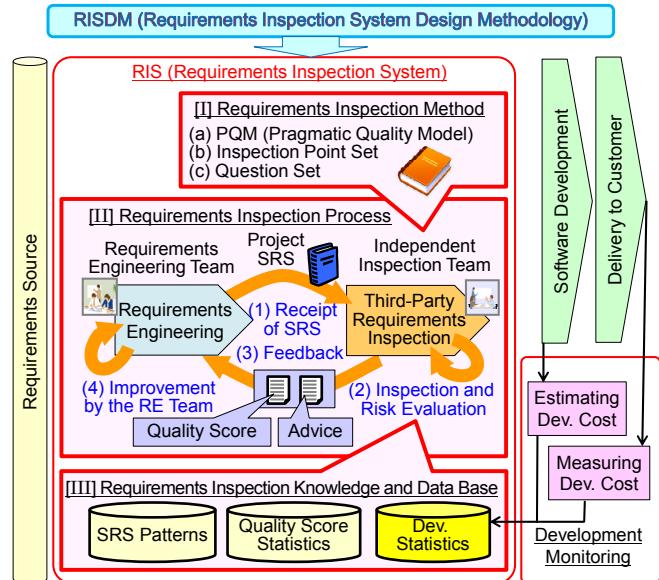


Fig. 1. RISDM and RIS for Third-Party Inspection

methodology for the RIS, which is assumed to apply to the third-party inspection process we proposed [26]. We assume two key stakeholders of RE (Requirements Engineering) teams and a third-party inspection team. The inspection team is responsible to inspect all the SRSes. The third-party inspection process consists of the following four processes.

1) *Receipt of SRS*: The inspection team receives an SRS from the requirements engineering team.

2) *Inspection and Risk Evaluation*: The inspection team inspects the SRS with the RIS, which should be designed in advance with the design methodology proposed. The inspection team compiles the inspection report, including QS (Quality Score) report and improvement advice. The QS report includes benchmarking of the QS of the SRS with QS statistics of other SRSes, and SRS patterns. Based on the QS, the team also evaluates the risk of the subsequent development. A set of advices is compiled to improve the quality of the SRS, and requirements engineering process of the team as well.

3) *Reporting with Evaluation and Feedback*: Based on the inspection report, the inspection team has a meeting with the RE team, and discuss possible improvements.

4) *Improvement done by the RE Team*: The RE team takes necessary actions based on the inspection report. If the quality of the SRS is concerned, the RE team revises the SRS and resubmit it to the inspection team for re-inspection.

The objective of this research is design methodology of RIS, a system of inspection. We use system for inspection since we view inspection as an organized and systematic activities to ensure the quality of the SRS. Therefore, we need to design the process and data model for the RIS just like design software systems.

B. Key Ideas for the RIS (Requirements Inspection System)

The basic idea of this work lies in that the requirements inspection is a system as a whole much like software development. Therefore, we can design the RIS (Requirements Inspection System), like we design a software engineering process and methodology with a reflection of “software process as a software”. As a design methodology of RIS, we propose

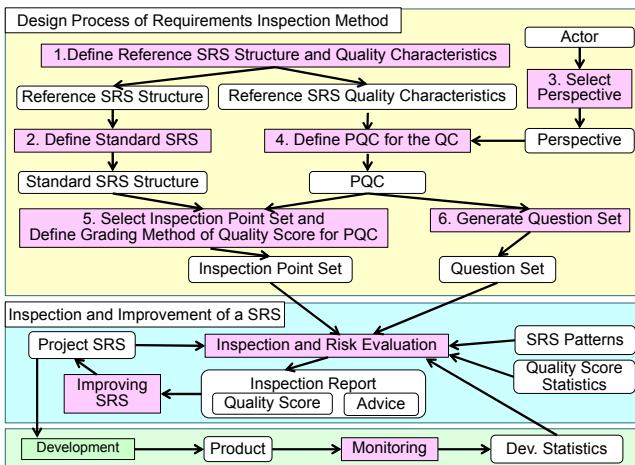


Fig. 2. Design Process of Requirements Inspection Method

RISDM (RIS Design Methodology), pronounced like rhythm, in this article.

As illustrated in Figure 1, the RIS consists of three key technologies indicated by three boxes:

- [I] Requirements Inspection Method,
- [II] Requirements Inspection Process, and
- [III] Requirements Inspection Knowledge and Data Base.

Requirements Inspection Method is a set of well-organized concrete techniques for the inspection. For the Requirements Inspection Method, we developed three key techniques including

- (a) PQM (Pragmatic Quality Model),
- (b) Inspection Point Set, and,
- (c) Question Set.

PQM is a comprehensive organization around PQC (Perspective Quality Characteristics), which is derived from Perspective and Reference QC (Quality Characteristics).

The core idea of Requirements Inspection Method comes from two ideas of PBR (Perspective-Based Reading) and Pragmatic Quality.

PBR helps to narrow down reading perspective to a specific perspective of the intended readers of the SRS. The Pragmatic Quality is an idea of quality for certain user of a product in a certain context. In the RIS, the pragmatic quality, we mean, a quality of the SRS for a user in the context of reading SRS. The PQC is a specialization of general QC, such as those defined in IEEE Std. 830. In this article, we propose a technique to derive PQC from the QC for certain Perspective, since the Perspective for PBR has to consistent with the Perspective to PQR. The Perspectives include customer/user and developer of the system to be developed.

Inspection Point Set is a set of points on the specific element of SRS to be inspected from a specific Perspective.

Question Set is a set of questions at the Inspection Point regarding on the each element of PQC.

By integrating the three techniques of PQM, Inspection Point Set, and Question Set. The Requirements Inspection Method enables the inspection team of little knowledge on domains to inspect diverse SRSes with productivity and quality of inspection.

Requirements Inspection Knowledge and Data Base provide information to the inspection team for inspection and evaluation. The Cost Statistics DB stores performance statistics of each project. It includes performance of both requirements engineering and subsequent development processes. As discussed later, we can take advantage of using the statistics for not only evaluate the quality of the SRS but also evaluate the risk of the subsequent development based on the quality of the SRS and statistics from the DB.

The results of the evaluation are compiled into the inspection report to the RE team. The inspection report also includes benchmarking of the quality of the SRS against the statistics of all the previous projects, and improvement advice with useful SRS patterns.

IV. DESIGN METHODOLOGY FOR RIS

A. Design Process of RIS

Figure 2 illustrates the design process of the RIS and execution of an instance of the RIS designed for inspection and improvement of Project SRSes in a specific project. The instance of the RIS is a part of the entire software development process as indicated in the bottom part of the figure.

By its definition, the design process, we propose, is expected to be executed before enacting the RIS. The design process consists of seven activities, and generates the PQM, the inspection point set, and the question set. An inspection team can systematically inspect an SRS by using the three techniques. The inspection team evaluates the quality of the SRS with QS (Quality Score) and compiles advices based on the QS. The QS and advices are compiled into an inspection report, and fed back to the RE team.

The inspection team refers the SRS patterns extracted from the best practice of SRSes, and statistics of QS. The RE team can improve the SRS based on the inspection report.

One of the uniqueness of the RIS system illustrated in Figure 1 is its capability to evaluate the risk of the subsequent development in terms of cost and schedule overrun based on the quality of SRS and development statistics collected.

B. Information Model of Requirements Inspection Method

Requirements inspection method is the heart of the RIS. Figure 3 illustrates an information model of the requirements inspection method corresponding to the design process illustrated in Figure 2.

There are three roots entities in the information model; Reference SRS Structure, Reference SRS QC (Quality Characteristics) and Perspective. The relationship of three root entities indicates the model of the requirements inspection method.

The Reference SRS Structure defines a TOC of SRS as a reference for the Project SRS to be inspected. The Reference QC defines generic quality characteristics for the SRS. The Perspective defined a specific view to the SRS for inspection. It might be a view from an actor, say a user or a designer. PQC (Pragmatic Quality Characteristics) is derived from the Reference QC for a specific Perspective.

The question set is a set of questions to inspect to an SRS

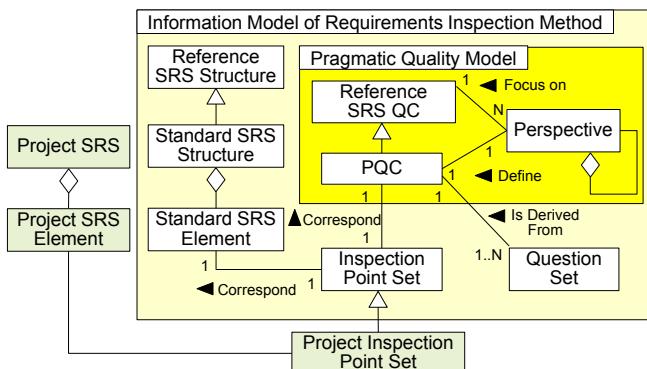


Fig. 3. Information Model of Requirements Inspection Method

from a perspective.

A standard SRS is derived from Reference SRS for a specific organization. Inspection point defines whether an inspection is necessary at a cross section between each element of the standard and each characteristic of PQC.

The Inspection Point Set is a collection of inspection points over all the elements and PQC. It is specialized to the specific SRS of the specific organization.

C. Design Process

To explain the details of the RISDM process, we employ real cases at NTT DATA and walk through the process.

1) *Define Reference SRS Structure and Quality Characteristics*: We select the Reference SRS Structure and Quality Characteristics as a baseline of the requirements inspection method. Actually, we employed IEEE Std. 830 as for both Quality Characteristics and SRS Reference Structure. It is possible to select any Quality Characteristics and SRS Reference Structure other than IEEE Std. 830.

2) *Define Standard SRS*: Based on the SRS Reference Structure selected in Activity (1), we define the Standard SRS Structure specific to the organization, say NTT DATA. If the organization already defined a Standard SRS Structure, it is only necessary to map the Standard SRS Structure and the Reference SRS Structure. The mapping ensures the completeness of the Standard SRS Structure.

Table 1 exemplifies the Standard SRS Structure of NTT DATA. The Standard SRS Structure includes TOC and a set of

TABLE I. TABLE OF CONTENTS OF STANDARD SRS

| Chapter | Section |
|--|---|
| 1. Introduction | 1.1. Purpose of SRS |
| | 1.2. Intended Reader |
| | 1.3. Structure of SRS |
| | 1.4. References |
| 2. Overview of System | 2.1. Goal of System |
| | 2.2. Business and Scope of System |
| | 2.3. Constraints |
| | 2.4. Terms |
| 3. Items Causing Change or Unspecified | 3.1. Items Causing Changes |
| | 3.2. Items Unspecified |
| 4. Functional Requirements | 4.1. Business Flow |
| | 4.2. Functions |
| | 4.3. Data Model Definition |
| 5. Non Functional Requirements | 5.1. NFR Grade |
| | 5.2. Requirements to System Architecture |
| | 5.3. Requirements to Migration |
| | 5.4. Requirements to Service Provisioning |

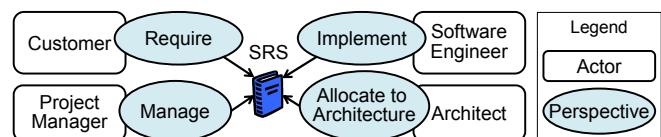


Fig. 4. Actors and Perspectives

TABLE II. PERSPECTIVES AND PQC

| Perspective | | Reference Quality Characteristics | | | | | | PQC (Pragmatic Quality Characteristics) | | |
|--------------------------|-------------------------------------|-----------------------------------|--|-------------|------------|-----------|------------|---|----------------------------|---|
| Level 1 | Level 2 | Correct | Ranked for importance and/or stability | Unambiguous | Verifiable | Traceable | Modifiable | ID | Name | Definition |
| Require | Conform to customer's needs | X | | | | | | C1 | Correspondence to goals | All business & system requirements should correspond to more than one project goal. |
| Manage | Manage all elements on the progress | | X | X | | | | C2 | Coverability | All contents of SRS should be described. |
| Implement | Design by template | | | X | X | | | C3 | Template usage | Template of SRS should be used. |
| | Design by standard description | | | X | X | | | C4 | Standard description usage | Standard description (notation) of SRS should be used. |
| | Use with common terms | | | X | | | | C5 | Definition of terms | Terms of the project SRS and their definitions are should be created. |
| Allocate to Architecture | Overview all the elements | | | | | | X | C6 | Listing with identifier | All artifacts should be stated in the artifact lists and have identifier. |
| | Specify all the elements | | | | | X | X | C7 | Identifiability | All artifacts and certain elements of the artifact are identifiable using identifier. |

associated SRS templates. The SRS templates include a template of functional descriptions and UI template. Table 1 shows only two levels of chapter and sections. However, The Standard SRS Structure specifies further details.

3) *Select Perspective*: Define a perspective for review in order to define the PQC concretely. First, identify an actor for whom the inspection team inspects the SRS. Then, define a set of perspectives to the Standard SRS from the actor(s) identified. If necessary, the perspective identified can be decomposed into detailed perspectives.

Figure 4 exemplifies four actors and associated perspectives we identified. Since the perspectives are top level, we call them Level 1 perspectives. The “Implement” and “Allocate to Architecture” perspectives are decomposed into detailed perspectives exemplified in the left column in Table 2.

4) *Define PQC for the QC*: Define PQC from the Reference QC (Quality Characteristics) in accordance with the Perspectives identified in Activity (3).

Table 2 exemplified the PQC. Here, we have four Perspectives at the top level. “Require” indicates a perspective from customer who require the requirements. It is refined to “Conform to customer's needs” at the second level of Perspective, Level 2. From the perspective, the “Correctness” QC needs to be assured, as indicated by “X” at the cross point of “Conform to customer's needs” perspective and “Correctness” QC. From the “Conform to customer's needs”, the “Correctness” QC is refined to “C1: Correspondence to goals” of PQC. Like this, all the QC elements are refined to PQC elements.

Note that, by the definition, the granularity of PQC is same to that of Perspectives, namely, seven Level 2 perspectives.

We call the structure of Perspective, Reference QC and PQC as PQM (Pragmatic Quality Model) as illustrated in Figure 3.

5) *Select Inspection Point Set and Define Grading Method of Quality Score for PQC*: This activity is to select inspection points set and define grading method of quality score for evaluating the SRS based on the PQC allocated to the SRS.

Inspection point is an element of SRS where an inspection is needed with respect a certain element of the PQC. The idea of inspection point comes from the fact that not every element of PQC is meaningful in the inspection to every element of the Standard SRS. Therefore, it is necessary to clarify which PQC element is effective in inspection to which element of the Standard SRS. Therefore, we select meaningful elements of PQC to each element of the Standard SRS.

The results of this selection are represented by “Inspection Point Set” illustrated in Table 3. In the table, each row indicates an element of PQC, and each column indicates an element of Standard SRS. The marking “X” at the cross point

TABLE III. INSPECTION POINT SET

| PQC | | TOC of Standard SRS | | | |
|-----|-------------------------|---------------------|--------------------------------|----------------|----------|
| ID | Name | 2.1 Goal of System | 2.2 Business & Scope of System | 2.3 Constraint | 2.4 Term |
| C1 | Correspondence to goals | X | | | |
| C2 | Coverability | X | X | X | X |
| C1 | Template usage | X | X | X | |
| C4 | Standard description | | X | | |
| C5 | Preparation of glossary | | | | X |
| C6 | Listing with identifier | X | X | X | |
| C7 | Identifiability | X | X | X | X |

is the inspection point which indicates inspection is effective and necessary. For example, “C1: Correspondence to goals”, an element of PQC, is effective and necessary to “Section 2.1 Goal of System”.

We identified 196 effective inspection points in our RIS.

Next, we need to define grading method of QS (Quality Score) for the inspection point set. QS is a score representing the quality of the SRS. Since we design every question, allocated to each inspection point, is simple and can be answered by “yes” or “no”, each inspection point can be scored to 1 or 0. Therefore, the quality of the SRS can be graded by summing the score over the inspection point set including 196 inspection points in our case.

The grading method is up to the organization that uses the RIS. However, we define the following basic grading method

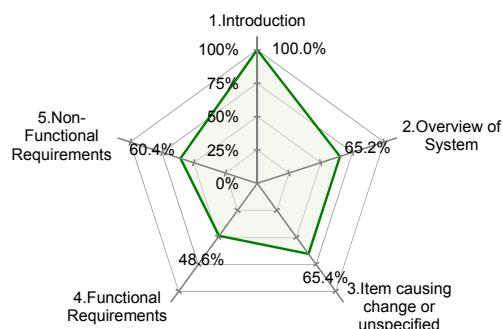


Fig. 5. An Example of Quality Score by Chapter

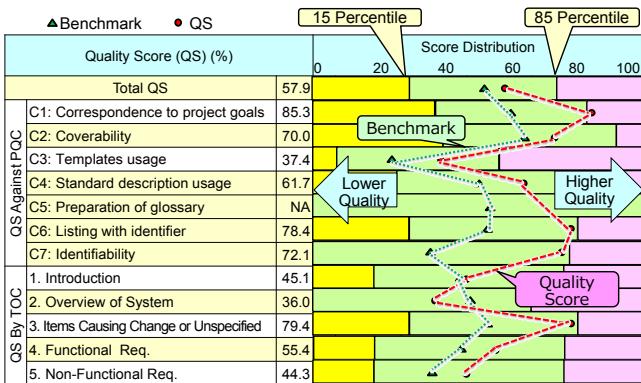


Fig. 6. Benchmarking with All Other Projects

based on the PQM.

a) Grading by PQC or Standard SRS Element: Sum of the score of the inspection by PQC or chapter of the SRS reveals the quality by PQC or chapter. We employ two types of grading.

i) TQS (Total Quality Score): By summing up the score and adjusting by the total number of the inspection points, we can evaluate an SRS by a simple scalar value of TQS ranging from 0 to 100.

ii) Distribution of PQC: As illustrated in Figure 5, a distribution of PQC shows the quality by chapter after adjustment with the baseline. A distribution by PQC illustrated in the upper half of Figure 6 indicates the strength and weakness of the SRS in terms of PQC.

b) Benchmarking: Figure 6 illustrates a benchmarking of an SRS against other project. The dotted line indicates the average score of all other projects and solid line indicated the score of the SRS. Note that the score is presented from two different views of PQC, C1 to C7, and chapter from 1 to 5.

6) Generate Question Set: Define “Question Set” from the PQC as exemplified in Table 4. Note that an element of “Question Set”, that is “a question”, corresponds to an element of PQC.

The question set also serves a precise reading guide, with which the inspection team reads the SRS from the specified perspective. Therefore, we think the question should be fine grained according to the specific PQC element, and make the reader clearly judge whether the specific SRS element meet the quality criterion in terms of the PQC element.

We employed a style of a question for checking an SRS element against a PQC element since it is simple enough to judge the quality of an element of SRS clearly while providing enough information on the SRS element.

V. PRACTICE

A. Corporate-Wide Application

The RIS designed by the proposed RISDM is practiced by a third-party inspection team in NTT DATA since 2010 after intensive empirical evaluation of the RIS. NTT DATA is a global software and service provider with some 10,000 professionals in Japan, and more than 60,000 worldwide [22].

TABLE IV. QUESTION SET

| PQC | | Question Set | No. of points |
|-----|----------------------------|--|---------------|
| ID | Name | | |
| C1 | Correspondence to goals | Are business and system requirements corresponding to the project goals? | 3 |
| C2 | Coverability | Are there elements of the project SRS corresponding to the elements of the standard SRS? | 54 |
| C3 | Templates usage | Are artifacts described using the template which is selected in the standard SRS? | 36 |
| C4 | Standard description usage | Are artifacts described by the standard description which is selected in the standard SRS? | 6 |
| C5 | Definition of terms | Are there glossaries of the project SRS created? | 3 |
| C6 | Listing with identifier | Are artifacts and certain elements of the artifact given identifier and listed in the table? | 48 |
| C7 | Identifiability | Are artifact and certain elements of the artifact identified using identifier? | 46 |
| | | | Total 196 |

As illustrated in Figure 7, the RIS is widely used by more than 140 projects in NTT DATA Japan. Now, it is one of the corporate wide standard processes for software development. The size of the projects inspected ranges from medium to large. Some 25% of the projects exceeds 500 person month of the efforts.

B. Practice of Third-Party Inspection Team

Within NTT DATA Japan, a third-party inspection team conducts the inspections based on the RIS. The team consists of three to four dedicated professionals and a few part time professional who can help the inspection team if the demand is too high. We provide training program for inspectors. It takes about one week for them to accquire the inspection method by the trainig program.

An inspection usually takes three weeks: one week for reading the SRS, the second week for analysis and evaluation of the SRS and quality score, and the third week for compiling an inspection report. The inspection team also monitors the development, and collect statistics.

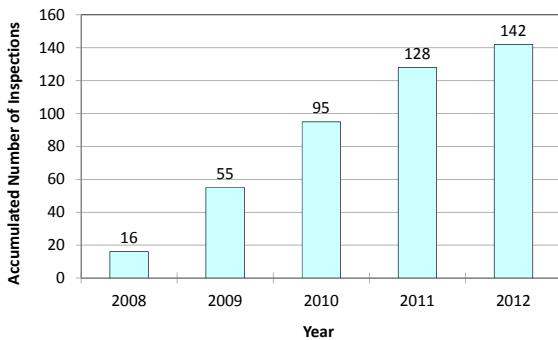


Fig. 7. Accumulated Numbers of Projects Inspected

TABLE V. PROJECT PROFILE FOR ANALYZING COST OVERRUN

| ID | Domain | Development Type | SRS pages |
|----|---------------|-------------------------|-----------|
| 1 | Financial | Extension & Replacement | 545 |
| 2 | Public | New | 401 |
| 3 | Manufacturing | New | 666 |
| 4 | Public | New | 172 |
| 5 | Financial | Extension & Replacement | 145 |
| 6 | Distributor | New | 300 |
| 7 | Service | New | 71 |

TABLE VI. CORRELATION COEFFICIENT AND P VALUE

| PQC | | Correlation Coefficient | P Value |
|-----|---------------------------------|-------------------------|---------|
| C1 | Correspondence to project goals | 0.516 | 0.236 |
| C2 | Coverability | -0.793 | 0.033 |
| C3 | Templates usage | -0.226 | 0.626 |
| C4 | Standard description usage | -0.219 | 0.638 |
| C5 | Preparation of glossary | -0.013 | 0.978 |
| C6 | Listing with identifier | -0.408 | 0.364 |
| C7 | Identifiability | -0.794 | 0.033 |

VI. EVALUATION

We found some useful characteristics of PQC in the practice of RIS designed by the proposed RISDM. To evaluate the RIS and associated PQM, we conducted an intensive analysis to statistics collected from the practice done by the third-party inspection team.

A. PQM for Risk Evaluation

1) *Purpose:* To demonstrate PQC is effective characteristics to predict risks of the subsequent development. The risk can be measured by cost, quality and time. We already identified impacts of the PQC to development time in terms of the delay of the delivery [25], and the impact of missing traceability. However, we do not have answers to the following questions.

- Is there any elements of PQC useful to predict any risks?, and
- Is there any elements of PQC which is more significant than others in predicting the risks such as cost overrun?

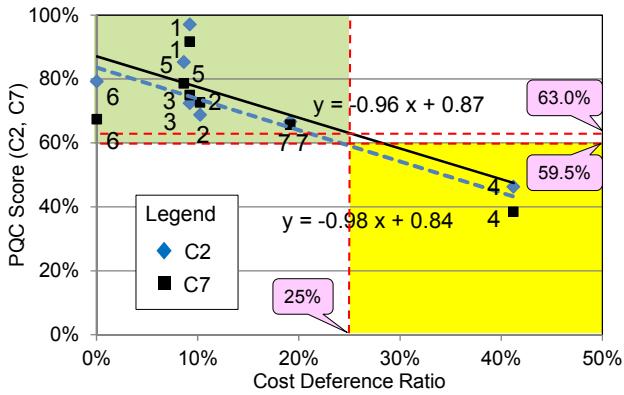
2) *Method:* As listed in Table 5, we collected statistics of seven projects, which are completed the entire development processes until 2013, and are complete in the sense that all the measures are completely measured throughout the development processes. The projects vary in domain, development type, and size in terms of number of pages of the SRS. The statistics include all the elements of PQC and cost of both estimated and actual the subsequent processes. The cost of subsequent development is estimated when the SRS is completed. The actual cost is fixed upon delivery. As illustrated in Figure 1, corporate wide project monitoring system collects the estimated and actual cost of each project and stores them to the development statistics DB. We referred the development statistics for the evaluation of CDR (Cost Difference Ratio), a ratio of estimated and actual cost, defined by the following equation (1).

$$CDR = \frac{|\text{Actual Cost} - \text{Estimated Cost}|}{|\text{Estimated Cost}|} \quad (1)$$

3) *Results and Findings:* Table 6 shows the results of correlation analysis between seven PQC of C1 to C7 and CDR. The table shows both correlation coefficient and P-value. It should be noted that the correlation coefficients of both C2 and C7 are less than -0.5, which indicates strong negative correlation with CDR. If the quality score of C2 or C7 of an SRS is low, the project has a high risk of cost overrun. Especially, P-value of C2 is less than 0.1, and the correlation stands to the population at the level of 10% statistical significance.

Figure 8 illustrates a scatter diagram between the PQC of C2 and C7 and CDR for six projects listed in Table 7. Linear regression for C2 and C7 are very closed. The diagram is separated into two parts by 25% of CDR. We use 25% threshold of CDR from COCOMO II [3].

The linear regression for C2 and C7 cross the 25% threshold of CDR at 59.5% and 63.0%, respectively. Therefore, we assume 59.5% is the lowest threshold of PQC, and further



number of pages of SRS was significantly increased from 106 to 191.

In the second revision, shown by dashed-dotted line, the C1, C2 and C7 are significantly improved. It means the quality of content, such as traceability, is semantically improved. Actually, the number of pages of the SRS was unchanged in this revision.

As a whole, both C2 and C7 are improved to attain the quality level of low risk.

From the detailed analysis, we found the process of how a project improved the quality of the SRS.

VII. DISCUSSIONS

A. Feasibility of RISDM

As discussed in the previous chapter, we believe the proposed RISDM is practically workable and useful. One of the contribution of the RISDM lies in that there is no design methodology for designing RIS, and many inspections, including RIS, are home grown and experience-based. Therefore, RIS is rather ad hoc, and differs project by project. Comparing with source code inspection, requirements inspection is much more difficult due to the diversity of its contents and writing styles. There is no consistent measure to measure the quality of SRSes across multiple projects.

Within the authors' knowledge, the proposed RISDM is first attempt to solve the problem. Although the practices are limited within a one company, the SRSes inspected exceeded 140, which are diverse in terms of size, domain, and writing styles. The success lies in the systematic design of the RIS with the proposed RISDM. Without the RISDM, we believe, it is still difficult to initiate a third-party inspection system for diverse applications.

It is worth to note that the RISDM is general enough to design different inspection system by changing the Perspective and PQC.

B. Effectiveness of PQM

Empirical study in the previous chapter discovered some nature of PQC (Pragmatic Quality Characteristics). Among them, we found two PQCs, Coverability (C2) and Traceability (C7) are influential to the quality of an SRS and subsequent software development process.

Poor coverability causes incompleteness of the SRS and makes the cost estimation inaccurate.

Poor traceability causes missing and/or misunderstanding.

If the evaluated score of C2 or C7 is low, it implies that risk of cost overrun significantly increased.

Therefore, by C2 and C7, we can predict the possible disaster, and help people to take any necessary strategy to avoid or mitigate the risks. This is particularly useful to the project manager and requirements analysts.

We also found that other PQC, such as "C4: Standard description usage" has less impact on the cost overrun of the project. However, such PQC may have an impact on other risks, which is not yet clarified.

C. Effectiveness of the Feedback

The in-depth analysis of the improvement of SRS, as illustrated in Figure 10, we found that the improvement of SRS is not uniform, and C2 and C7 are considered as key indicators to evaluate the improvement.

On the other hand, we found that 60% of the average of C2 and C7 is a lower threshold of the quality level as indicated in Figure 9. The fact is discovered by this study, and RE teams are not aware it. However, all the RE teams considered 60% as the lowest goal of the quality. This implies that improvement activities of the RE teams are reasonable and good enough.

It is worth to note that the question set is useful for an improvement guideline, although it is designed for evaluate the SRS.

D. Integrated Methodology

The RISDM is the design methodology for integrating pragmatic quality and PBR (Perspective Based Reading) for inspecting SRS. This methodology provided the systematic design process of requirements inspection method. It enables even third-party inspectors, who have less knowledge on project context and background, can conduct an inspection of project SRSs. With reference to the method defined by the RISDM, in NTT DATA, the third-party inspection process for SRSs of large scale software development projects could work in practice. As above mentioned, two approaches (i.e., pragmatic quality, PBR) have been studied independently. Each approach could make some contribution, but is limited to partial solution. Our technical contribution is that we have shown the integration of both approaches could elaborate a practical and effective methodology, namely RISDM. Moreover, the RISDM is unconstrained by existing domain knowledge. It could be applied to not only enterprise system development, but also embedded system, and package-based solution.

VIII. CONCLUSIONS AND FUTURE WORKS

This paper proposed RISDM (Requirements Inspection System Design Methodology), a methodology for designing the RIS (Requirements Inspection System). Our methodology provides a design process of requirements inspection method by integrating both approaches of pragmatic quality and PBR (Perspective Based Reading). In the process, we defined the PQM (Pragmatic Quality Model) which is composed of Perspective of certain reader of SRS, Reference QC (Quality Characteristic) of SRS, and PQC (Pragmatic Quality Characteristics) which are derived from Perspective and Reference QC. Next, the question set is generated from the set of PQCs. The question set makes the inspector clearly judge whether the specific elements of the project SRS meet the quality criterion in terms of the PQC element. We have designed and practiced the inspection method to an industrial inspection process. Moreover, after conducting the inspection of the project SRS, we have monitored the life cycle of the project and collected data on SRS improvement activity and development cost. From the statistical analysis of the data, we found the effectiveness of the PQM for risk evaluation in terms

of cost overrun. We also found the effectiveness of improving the SRS by the advice which is derived from the inspection by applying the question set to the SRS. Within our knowledge, this work is the first contribution of proposing a design methodology of inspection system which enables to inspect the SRS systematically.

In future work, we plan to analyze a detailed relation between the PQC and the difference of the cost, quality and delivery-time for risk evaluation with high accuracy. We will also apply the proposed methodology to another type of development project, including embedded system, and package-based solution.

ACKNOWLEDGMENT

The authors are grateful to Dr. T. Kitani, Dr. T. Hayama, Mr. T. Kobashi, Mr. M. Hiraoka, Ms. M. Fujinuki, Mr. A. Namikawa, Dr. N. Ohsugi, and Mr. H. Yamamoto of NTT DATA Corporation for their assistance.

REFERENCES

- [1] I. F. Alexander and R. Stevens, *Writing Better Requirements*, Addison-Wesley, 2002.
- [2] B. W. Boehm, "Verifying and Validating Software Requirements and Specifications," *IEEE Software*, Vol. 1, No. 1, Jan/Feb. 1984, pp. 75-88.
- [3] B. W. Boehm, et al., *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000.
- [4] B. H. C. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering," *FOSE '07*, IEEE Computer Society, May 2007, pp. 285-303.
- [5] D. Damian and J. Chisan, "An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management," *IEEE Trans. Software Engineering*, Vol. 32, No. 7, Jul. 2006, pp. 433-453.
- [6] A. Davis, et al., "Identifying and Measuring Quality in a Software Requirements Specification," *Proc. of the First Int'l Software Metrics Symposium*, IEEE Computer Society, May 1993, pp. 141-152.
- [7] C. Denger and T. Olsson, "Quality Assurance in Requirements Engineering," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Springer, 2005, pp. 163-185.
- [8] C. Denger and F. Shull, "A Practical Approach for Quality-Driven Inspections," *IEEE Software*, Vol. 24, No. 2, Mar./Apr. 2007, pp. 79-86.
- [9] F. Fabbrini, et al., "Achieving Quality in Natural Language Requirements," *Proc. of 11th Int'l Software Quality Week Conference (QW'98)*, May 1998, 17 pages.
- [10] M. E. Fagan, "Advances in Software Inspections," *IEEE Trans. Software Engineering*, Vol. SE-12, No. 7, Jul. 1986, pp. 744-751.
- [11] K. Fosberg and H. Mooz, "System Engineering Overview," R. Thayer and M. Dorfman (eds.), *Software Requirements Engineering*, IEEE, 2000, pp. 44-72.
- [12] G. Fanmuy, et al., "Requirements Verification in the Industry," *Proc. of CSDM 2011*, Springer, 2012, pp. 145-160.
- [13] G. Génova, et al., "A Framework to Measure and Improve the Quality of Textual Requirements," *Requirements Engineering J.*, Vol. 18, No. 1, Mar. 2013, pp. 25-41.
- [14] T. Gorschek and A. M. Davis, "Requirements Engineering: In Search of the Dependent Variables, *Information and Software Technology*, Vol. 50, No. 1-2, Jan. 2008, pp. 67-75.
- [15] IEEE Std. 830-1998, *Recommended Practice for Software Requirements Specifications*, IEEE, 1998.
- [16] IEEE Std. 1028-1997, *Standard for Software Reviews*, IEEE, 1997.
- [17] JISA, Requirements Engineering Body Of Knowledge (REBOK), Ver. 1.0, Kindai Kagakusha, 2011 (In Japanese).
- [18] E. Kamsties, et al., "Detecting Ambiguities in Requirements Documents Using Inspections," *Proc. of WISE'01*, 2001, pp. 68-80.
- [19] E. Knauss, et al., "Investigating the Impact of Software Requirements Specification Quality on Project Success," *Proc. of PROFES '09 LNBI* Vol. 32, Jun. 2009, pp. 28-42.
- [20] J. Krogstie, et al., "Towards a Deeper Understanding of Quality in Requirements Engineering," *Proc. of CAiSE '95*, LNCS Vol. 932, Springer, 1995, pp. 82-95.
- [21] C. J. Neill and P. A. Laplante, "Requirements Engineering: The State of the Practice," *IEEE Software*, Vol. 20, No. 6, Nov./Dec. 2003, pp. 40-45.
- [22] NTT DATA, <http://www.nttdata.com/global/en/>.
- [23] K. Pohl, *Requirements Engineering*, Springer, 2010.
- [24] A. Porter and L. G. Votta, "An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspections," *Proc. of ICSE 1994*, IEEE Computer Society, pp. 103-112.
- [25] S. Saito, et al., "Empirical Analysis of the Impact of Requirements Traceability Quality to the Productivity of Enterprise Applications Development," *Proc. of APSEC 2012*, IEEE Computer Society, Dec. 2012, pp. 330-333.
- [26] S. Saito, et al., "Requirements Clinic: Third Party Inspection Methodology and Practice for Improving the Quality of Software Requirements Specifications," *Proc. of RE 2013*, IEEE Computer Society, Jul. 2013, pp. 290-295.
- [27] F. Salger, et al., "An Integrated Quality Assurance Framework for Specifying Business Information Systems," *Proc. of CAiSE Forum 2009*, Jun. 2009, pp. 25-30.
- [28] F. Salger, Requirements Reviews Revisited: Residual Challenges and Open Research Questions, *Proc. of RE 2013*, IEEE Computer Society, Jul. 2013, pp. 250-255.
- [29] F. Shull, et al., "How Perspective-Based Reading Can Improve Requirements Inspections," *IEEE Computer*, Vol. 33, No. 7, Jul. 2000, pp. 73-79.
- [30] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, Wiley, 1997

Tackling the Requirements Jigsaw Puzzle

Maria Pinto-Albuquerque

School of Technology and Architecture,
ISCTE-IUL (Inst. Univ. Lisboa), Portugal, and
School of Computing and Communications,
Lancaster University, UK
maria.albuquerque@iscte.pt

Awais Rashid

School of Computing and Communications,
Lancaster University, UK
marash@comp.lancs.ac.uk

Abstract—A key challenge during stakeholder meetings is that of presenting the requirements and conflicts to stakeholders in a way that fosters co-responsibility and co-ownership regarding the conflicts and their resolution. In this paper, we propose a jigsaw puzzle metaphor to make identified conflicts explicit as well as an associated method to utilise this metaphor during stakeholder meetings. The metaphor provides an easy to understand language for stakeholders from otherwise diverse backgrounds. It enables stakeholders to work with a well-understood concept - that of building a system from misshapen pieces. These characteristics foster communication and team work, which improve commitment of stakeholders in co-authoring of requirements and co-responsibility in conflict handling. The gamification of conflict resolution also promotes a relaxed environment, which in turn improves team cooperation and creativity. Our experience in three user studies demonstrates that the jigsaw puzzle indeed improves such co-responsibility and co-ownership when compared with typical text-based representations of requirements.

Index Terms—Requirements, conflict, creativity, game, jigsaw puzzle, stakeholders, team work, communication, metaphor, visualization.

I. INTRODUCTION

Although brainstorming, one of the most popular creativity techniques used for requirements identification, dates back to 1935, the use of creativity techniques in RE is still under investigated [25]. In this paper, we focus on fostering creativity with regards to resolving requirements conflicts during stakeholder meetings. Such meetings often take place between analysts and customer stakeholders to inspect and/or negotiate conflicts in requirements, and to update requirements as needed. A key issue during such meetings is that of explicitly presenting the requirements and the conflicts to the stakeholders in a way that fosters co-responsibility and co-ownership.

Fostering such co-responsibility and co-ownership is challenging for a variety of reasons. Firstly, though methods have been developed to provide decision support for identifying and handling ambiguity and inconsistency, these methods “code” such information (that depicts the conflict) in a technical formalism. Stakeholders, however, often possess heterogeneous backgrounds. Current formalisms (e.g. [6, 11, 19, 27]) are not readily accessible to users with non-computing backgrounds (and even different cultural backgrounds in current multi-national settings can pose a challenge in this regard). Secondly, studies [20] have shown that analogical

reasoning with unfamiliar domain classes is difficult without prior learning. Finally, RE is not only an analytical task but also a creative one. The use of too many analogies during stakeholder meetings can hamper analogical reasoning [21, 22] and hence creativity.

Of course, requirements conflicts arise for a variety of reasons. On the one hand, the vast majority of requirements documentation is described using natural language [24]. As natural language is inherently imperfect [9, 23], requirements may contain conflicts due to incompleteness, ambiguity, or simply the definition of the same feature in two incompatible ways (inconsistency) [3]. On the other hand, stakeholders often have different interests. These interests/desires may be technically impossible, may place other requirements at risk or lead to increased costs [33]. In this paper, we are not concerned with the root causes of conflicts. Instead, our focus is on how to present these conflicts to stakeholders during consultation meetings in a fashion that enables them to collectively recognize the challenges posed by the conflicts and the obstacles that have the potential to compromise the project as a whole, and its quality.

It has been argued that a lack of specialist notation improves communication in multi-disciplinary contexts [26]. Given the typical diversity of stakeholder backgrounds, we therefore focus on communicating conflicts through a “language” that is easy to understand and relate with. We propose a jigsaw puzzle game to make identified conflicts explicit. In this jigsaw puzzle game, each puzzle piece represents a requirement. When the requirement text contains conflicts with other requirements, the respective puzzle pieces almost fit together but not perfectly. The jigsaw puzzle metaphor provides an easy to understand language for stakeholders from otherwise diverse backgrounds. Furthermore, it enables stakeholders to work with a well-understood concept – that of building a system from misshapen pieces. These characteristics foster communication and team work, which improve commitment of stakeholders in co-authoring of requirements and co-responsibility in conflict handling. The gamification of conflict resolution also promotes a relaxed environment, which in turn improves team cooperation and creativity.

Our experience in three different user studies shows that the jigsaw puzzle indeed fosters such team work and communication compared with the use of typical textual presentations. It also improves the commitment of stakeholders

in the co-authoring of requirements and co-responsibility in conflict handling.

The novel contributions of this work are as follows:

1) A visual metaphor in the form of a jigsaw puzzle to communicate requirements conflicts to stakeholders. Though work exists on use of information visualisation [10, 2, 36], games [18, 31] and creativity techniques [22, 25] during requirements engineering, such a metaphor and game has not been considered to date. Given the easy to understand and gaming nature of the metaphor, there is little or no need to introduce the metaphor (as we found in our user studies). The meeting itself is a game, a jigsaw puzzle, which is at the same time the conflict communication mechanism and the means to foster discussions regarding conflict resolution strategies.

2) Our use of a jigsaw puzzle metaphor leads to a separation of the processing of the information about the conflicts from the issue of communicating those conflicts. This separation is crucial to permit a user-centred design of the communication of the requirements and its conflicts. Such a separation, though critical, has not been proposed to date.

The remainder of this paper is organized as follows: Section II summarizes the related research. Section III describes the jigsaw puzzle game, while Section IV explains how the jigsaw puzzle should be used in the stakeholder consultation meetings. Section V presents the evaluation of the approach. Section VI discusses what the potential impact of the insights from the jigsaw puzzle game hold for the state-of-the-art in stakeholder consultation meetings and the interactions between requirements engineers and stakeholders, as well as the future work we envision. Finally, Section VII presents the conclusions.

II. RELATED WORK

A. Requirements Elicitation and Creativity

Maiden et al. have developed a RE process, RESCUE [22], which incorporates creativity workshops to foster creative thinking to discover requirements. These creativity workshops use several techniques to encourage creativity, including brainstorming and analogical reasoning [20, 21, 22]. Their work demonstrates that: 1) Workshop participants found it difficult to exploit the analogies (e.g., between air traffic management and textile and music domains) [20, 21]; and that strategies are needed to make analogical reasoning more effective [22]. 2) Brainstorming generated more creative ideas than analogical reasoning, and it was more cost-effective and easier to use. However the ideas obtained with analogical reasoning were described in more detail than the brainstormed ideas [21].

The jigsaw puzzle game aims to address the issue of analogies that are difficult and costly to explain and learn. It provides two easy to understand visual analogies. One is the analogy between a jigsaw puzzle and a system both built out of pieces to create a desired object. This analogy reinforces the view of RE as creative task, not only analytical one. The other

analogy employed uses a misshapen graphical visualization of a puzzle piece to represent that there is a problem.

Games are often used at the beginning of elicitation meetings, to induce a fun and relaxed environment, and promote creativity [22]. In this approach there is no need to introduce a game; the meeting itself is a game, a jigsaw puzzle, which is also the communication mechanism.

B. Visual Communication Metaphors in RE

Gotel et al. [14] envisioned the potential synergy between Information Visualization and SE Visualization to provide good metaphors for software development tools, enabling effective communication in software development. Visual metaphors have, for a long time, been used to represent information in SE. In the last fifteen years the predominant visualizations in the area of RE are either associated with UML diagrams or i* goal models [14]. Diagrams use geometric shapes and geometric relations to build a visual language with defined syntax and an associated semantics. These visualizations present apprehension problems (i.e., the structure and content of visualization is not readily perceived and comprehended [35]). Thus, they need to be learned. Field studies [5] confirm that the visualization's efficacy in diagrams is dependent on the user's previous experience and visual literacy, which can present a barrier for communication and work among multidisciplinary user profiles, and thus between requirements engineers and stakeholders. Some more sophisticated visual metaphors have been recently developed to visualize software (e.g. [10, 2, 36]), but with different goals compared to this work. Boccuzzo et al. [4] describe an interesting usage of the concept of well-shaped graphical visualization to represent that the corresponding artifact is well designed. Their work shows that the metaphor "language" can be used to express information (quality aspects) about the artifacts that are being represented. Boccuzzo et al.'s work inspired us to develop the analogy that uses a misshapen graphical visualization of a puzzle piece to represent that there is a problem.

C. Inconsistency and Ambiguity Handling

Inconsistency identification has been recognized as a RE problem for many years. Viewpoints and Inconsistency Management (VIM) [13, 28, 11] and the Non-Functional Requirements Framework (NFRF) [6] provide specification formalisms that explicitly represent and manage relationships between the artifacts (ViewPoints, Softgoals respectively) that represent the knowledge about the system to be built. However, one needs to be knowledgeable in the formalism (and these are quite complex) to be able to manage the conflict relationships. The inconsistency detection tools (e.g. [30]) developed under the umbrella of aspect-oriented textual requirements approaches that use semantics-based decomposition present the output as a list of possible inconsistency situations. These tools, however, do not address the communication goals (in particular with non-SE stakeholders), which are the focus of the jigsaw puzzle game.

A number of techniques also exist to reduce the level of ambiguity in requirements (see e.g., [3]). Our approach aims to

use the outputs of ambiguity and inconsistency detection tools as input to build the jigsaw puzzle game, making it a communication mechanism accessible to all kinds of stakeholders, thus promoting their participation and aiming at co-responsibility in requirements. A contribution of our work is heuristics to identify ambiguities and inconsistencies that may need to be flagged using the jigsaw puzzle game (see sections III.A and III.B).

III. JIGSAW PUZZLE GAME

In the jigsaw puzzle metaphor each puzzle piece represents a requirement. When the requirement's text potentially leads to conflicts with other requirements, the respective puzzle pieces have a matching edge that almost fits but not perfectly. The background of each jigsaw puzzle piece contains part of a picture. When all the pieces of a puzzle are assembled correctly each piece contributes to a bigger complete image, just like in commonly used jigsaw puzzles. The jigsaw puzzle game uses the cues commonly used in jigsaw puzzle pieces: pieces that have straight borders, and thus belong to corners and jigsaw borders; interlocking shapes that match pairs of pieces; and an image in the background, with each piece containing a part of it. Another cue is the orientation of the text of the requirements.

In order to demonstrate an example, we utilise the Non-functional requirements (NFRs) from the Crisis Management Systems (CMS) requirements documentation [17]. We note that this is a case study available in literature and any inconsistencies and ambiguities in the requirements exist in the original documentation and have not been introduced for the purposes of demonstrating the approach in this paper.

Figure 1 shows a picture of the four pieces of such a jigsaw puzzle representing the NFRs Availability, Reliability, Real-time, and Accuracy.

Figure 2 and 3 present the text of the Real-time and the Availability requirements, respectively. The text written in a jigsaw piece representing a requirement does not capture the full text as in the requirements documentation. Instead the goal is to capture essential elements in particular due to the need to improve readability. Some unnecessary text is cut out, some words are abbreviated, the text is displayed in list mode, and upper case letters are used to stress the "topic" of each requirement.

The interlocking shapes between the pieces in Figure 1 do not match perfectly, depicting conflicts between the requirements, represented by the pieces. For instance the bad fitting between the Availability piece and the Real-time piece represents the following inconsistency:

The Real-time requirement has the word expressions: "receive and update...information", "communication of information", and "retrieve any stored information". For these situations to happen, the system must be available, thus they are related to the following word expressions of Availability: "in operation except for a maximum downtime" and "recover...upon failure". The possible inconsistency situation resides in the following. If the system is allowed a downtime for 2 hours (as allowed by the Availability requirement), how can it guarantee the Real-time requirement, i.e., "receive and update...information...at intervals not exceeding 30 seconds", "the delay in communication of information..shall not exceed 500 milliseconds", and "retrieve any stored information with a maximum delay of 500 milliseconds".

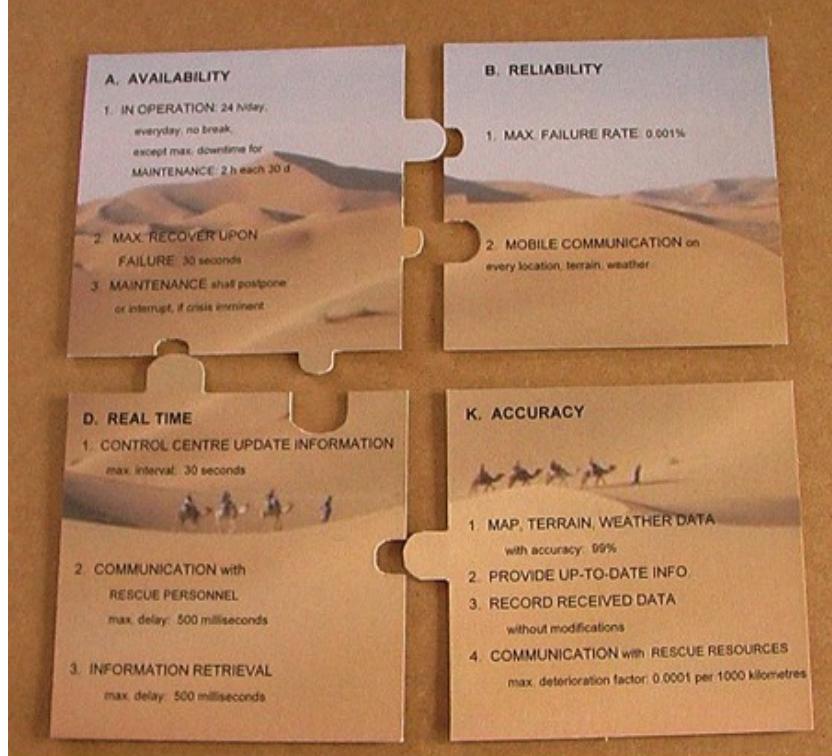


Fig. 1 The jigsaw puzzle for the requirements Availability, Reliability, Real-time, and Accuracy of the CMS case study

The same type of question can be posed in relation to the second phrase of the Availability requirement, which states: “The system shall recover in a maximum of 30 seconds upon failure”. But, thus if it may be up to 30 seconds down how can the system guarantee what the Real-time requirement demands?

- Real-time
 1. The control centre shall receive and update the following information on an on-going crisis at intervals not exceeding 30 seconds: resources deployed; civilian casualties; crisis management personnel casualties; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; estimated time of arrival (ETA) of rescue teams on crisis site.
 2. The delay in communication of information between control centre and rescue personnel as well as amongst rescue personnel shall not exceed 500 milliseconds.
 3. The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.

Fig 2. Text for the Real-time requirement [17]

- Availability
 1. The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance.
 2. The system shall recover in a maximum of 30 seconds upon failure.
 3. Maintenance shall be postponed or interrupted if a crisis is imminent without affecting the systems capabilities.

Fig 3. Text for the Availability requirement [17]

A jigsaw puzzle game, such as the one in Figure 1, can be prepared using a machine but, for the purpose of this research it was just prepared by hand (though the pieces were generated using software implemented in MATLAB, they are printed and cut manually). In the longer term such a process can be fully automated though we note that, in consultation meetings with stakeholders, typically, requirements engineers want to focus on a small number of requirements (say 4 to 6) [33]. Therefore, the jigsaw puzzle needn’t contain a large number of pieces so as not to detract from the key conflicts that should be focus of discussion. Other permutations are possible such as using touchscreen interfaces or augmented reality though they would lose some of the tactile and physical manipulation aspects of the jigsaw puzzle that were welcomed by users during our studies.

In order to generate the bad-fitting pieces it is necessary to identify the most pertinent or most problematic conflicts that should be the focus of the meeting. As we noted above, there is a significant body of work on ambiguity and inconsistency identification as well as tools, which can be used for the purpose. We, therefore, build on this body of work (as well as our own insights and experience) to provide suitable heuristics to identify sources of ambiguity or inconsistency that may need to be flagged using the jigsaw puzzle. We focus on both ambiguity and inconsistency identification heuristics. This is

because the definition of ambiguity as being a property of both the text and the interpretations held by a group of readers of the text [37] enables to establish a connection between ambiguity and relations among requirements with unknown influence but with a potential to be an inconsistency. In such cases, the analysis by the stakeholders of the two pieces of the SRS (software requirements specification) to decide if there is an inconsistency promotes the clarification of the interpretation each person gives to the parts of the SRS in appreciation. This clarification of the interpretation(s) is what is needed to resolve the ambiguity, or to boost the discussion on the ambiguity itself. It can happen the other way round, i.e., when analyzing ambiguity in two parts of the SRS, the resolution of the ambiguity may clarify if certain relations between the requirements in those two parts of the SRS are inconsistent.

As our focus is on resolving conflicts amongst NFRs, often the heuristics below advise the search for word expressions referring to qualities or breaking of qualities. The domain of qualities to be considered for such search can be taken from catalogues of qualities. Such catalogues have been developed in the NFRF [6], with developments like the work of Cysneiros et al. [8]. These catalogues may also be standards such as the ISO/IEC 25010:2011 [16]. For each heuristic, we provide both the heuristic and an example. It is important to notice that the final selection of the sets of requirements, to be used in a group consultation meeting, belongs to the team of requirements engineers undertaking the RE process. The heuristics are a means to support the engineers’ work in identifying the requirements that should be the focus of the jigsaw puzzle game.

A. Identification of the Most Problematic Ambiguities

We propose the following heuristics to identify ambiguity¹. They draw upon the definition of an unambiguous SRS [15], and the classification of ambiguity proposed by Berry et al. [3], and applies them to the domain of NFRs. Ambiguity is classified into: lexical, syntactic (or structural), and semantic.

1) *Lexical Ambiguity Heuristic*: Berry et al. define: “lexical ambiguity occurs when a word has several meanings” [3]. *Heuristic* - To find ambiguity in a SRS:

1. search for:
 - word expressions that belong to a catalogue of qualities (or their synonyms); e.g.: accurate; or
 - word expressions meaning a violation of quality (including antonyms of the qualities); e.g.: failure; and
2. check whether the word expression may have more than one meaning, enabling more than one interpretation.

Example. “Are control centre and system, referred in Real-time requirement, the same or different entities?” It is not clear if the expressions ‘control centre’, and ‘system’ refer to the same entity. There are at least two possible interpretations: one that ‘control centre’ and ‘system’ are the same entity; and the other that they are different entities. Such pairs of words constitute ambiguity cases quite often related with inconsistency.

¹ “A SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation” [15].

2) *Syntactic Ambiguity Heuristic*: Berry et al. define: “syntactic ambiguity, also called structural ambiguity, occurs when a given sequence of words can be given more than one grammatical structure, and each has a different meaning” [3].

Heuristic - To find ambiguity in a SRS:

1. search for:
 - sequence of words referring to a quality from a catalogue (or its synonyms); or
 - sequence of words referring to a violation of a quality from a catalogue (including antonyms of the qualities); and
2. check whether that sequence of words can be given more than one grammatical structure, each having a different meaning, enabling more than one interpretation on how that quality or breaking of quality has to be realized in the system.

Example. “Is this a list of five types of interaction modes, or just three?” The CAS system SRS [1] describes: “SaaS applications offered through the Internet are usually supporting different interaction modes including classic page-oriented HTML GUIs, rich internet GUIs (e.g., AJAX) as well as access channels for mobile users such as WAP, data replication for offline use or speech control”. Different interpretations are possible depending on diverse assumed grammatical structures. It is possible to interpret that the phrase lists five “types of interaction modes”: a) classic page-oriented HTML GUIs, b) rich internet GUIs, c) channels for mobile users such as WAP, d) data replication for offline use, e) data replication for speech control. Or that there are just three types of interaction modes: a) classic page-oriented HTML GUIs, b) rich internet GUIs, and c) channels for mobile users.

3) *Semantic Ambiguity Heuristic*: Berry et al. define: “semantic ambiguity occurs when a sentence has more than one way of reading it within its context although it contains no lexical or syntactic ambiguity” [3]. We adopt the definition of context by Berry et al: “The SRS context comprises the language context (i.e. the sentences before and after the sentence in which the quality word expression occurs) and the context beyond the language (i.e. the situation, the background knowledge, and expectations of the speaker or hearer and the writer or reader)” [3].

Berry et al. define: “semantic ambiguity occurs when a sentence has more than one way of reading it within its context although it contains no lexical or syntactic ambiguity” [3]. We adopt the definition of context by Berry et al: “The SRS context comprises the language context (i.e. the sentences before and after the sentence in which the quality word expression occurs) and the context beyond the language (i.e. the situation, the background knowledge, and expectations of the speaker or hearer and the writer or reader)” [3].

Heuristic - To find ambiguity in a SRS:

1. search for:
 - word expressions that belong to a catalogue of qualities (or their synonyms); or
 - word expressions meaning a violation of a quality (including antonyms of the qualities); and

2. check whether the sentences using these word expressions have more than one way of reading it within the SRS context, enabling more than one interpretation on how that quality or breaking of quality has to be realized in the system.

Example. “Is downtime for maintenance in Availability requirement to be accounted for failure rate referred in Reliability requirement?” The first sentences of the Availability (see Figure 3) and Reliability requirements from the CMS document show an example of semantic ambiguity. Considering that each of these sentences is part of the context of the other; and considering also the background knowledge and the expectations of the readers, there are two ways of reading the following two sentences. From the Availability requirement: “1. The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance”, and from the Reliability requirement: “1. The system shall not exceed a maximum failure rate of 0.001%”. These two ways of reading are: 1) what is required in the second sentence concerning the ‘maximum failure rate of the system’ has to take into consideration the ‘maximum downtime’ allowed in the first sentence or; 2) these concepts are not related to each other. The ambiguity can be expressed with the question: should ‘downtime for maintenance’ be included in the situations considered for the calculation of ‘failure rate’, and thus the times allowed for ‘downtime’ have to be accounted in the ‘failure rate’ value; or ‘downtime ... for maintenance’ is not to be accounted for in the ‘failure rate’?

B. Identification of the Most Problematic Inconsistencies

We propose the following heuristics to identify inconsistencies. They draw upon the definition of an internally consistent SRS², and our own insights inspired from existing RE approaches namely: Preview [34], and the NFRF[6].

1) *Synonymous/Antonymous Quality Inconsistency Heuristic*: When searching for pairs of NFRs defining features (qualities, in this case) of the system in an incompatible way, one should search for two phrases in the SRS referring to the same quality. Picking a quality ‘X’ referred with the word expression ‘X1’, other reference to the quality ‘X’ can be done using a synonym of ‘X1’; or in a negative form, using an antonym of ‘X1’. Preview [34] uses the concept of “focus” to find probable conflicts. Viewpoints whose foci intersect are the most likely sources of conflict. In fact these viewpoints with intersecting foci are the ones that impose requirements on the same system components or features, and thus the ones where conflicts are more likely to appear. When for a quality ‘X’ referred with a word expression ‘X1’, an antonym or synonym (say ‘X2’) of ‘X1’ is found in another part of the SRS, these two parts of the SRS are describing the same quality (i.e. impose requirements on the same features) and

²Using the IEEE Std. 830-1998 [15] and the definition of contradiction by Meyer [23] as a basis, we define that: A SRS is internally consistent if, and only if, no subset of individual requirements described in it define a feature of the system in an incompatible way.

thus they may conflict. In the Preview approach such a pair ('X1','X2') would belong to two viewpoints that have intersecting foci, and thus would be selected as probable conflicts.

Heuristic - To find an inconsistency situation in a SRS:

1. search two word expressions from a catalogue of qualities (or violation of qualities), which are synonym or antonym of each other, and
2. check whether their descriptions are incompatible with each other.

Example. “Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement”. The possible inconsistency situation has to do with the fact that the Real-time requirement describes that “The control centre shall receive and update the following information ... at intervals not exceeding 30 seconds.” Thus it offers some detail about the “update of information”. But the Accuracy requirement just says: “The system shall provide up-to-date information...” The issue is: “are these requirements compatible?” and “How can they be made compatible?”

2) *Actions Operationalizing Quality Inconsistency Heuristic:* In computer science and engineering it is known that an abstract concept such as a quality (e.g. Real-time) has to be operationalized in less abstract concepts usually described by actions (e.g. receive and update information). The description of the abstract concept and its operationalizations should be compatible. In Preview it is sometimes helpful to consider the decomposition of a viewpoint into sub-viewpoints. This is advised in the presence of conflict among the requirements of a viewpoint, especially if the sources of the requirements have imperfectly matched foci. This heuristic (as the Quality dependency inconsistency heuristic described below) aims to identify these cases of conflict among requirements of a viewpoint.

Heuristic - To find an inconsistency situation in a SRS:

1. search two word expressions of actions describing the operationalization of the same quality (from a quality catalogue), and
2. check whether these descriptions are incompatible with each other.

Example. “Real-time description requiring times possibly incompatible.” In the Real-time requirement, described in Figure 2, it is questionable if the time intervals and limits described are compatible among them.

3) *Quality Dependency Inconsistency Heuristic:* It is pertinent to organize system qualities in hierarchies, with more general qualities at the top and more specific ones at lower levels of the hierarchy. ‘Availability’ is certainly a quality belonging to the top level. If a system is not available it is useless to discuss other qualities. At a lower level of abstraction it is useful (although arguable on how to do it best) to organize, for instance, in a tree below ‘Performance’: ‘speed’, ‘efficiency’, ‘resource consumption’, ‘throughput’, ‘response time’. Such hierarchies can be found in standards (e.g. [16]). The NFRF [6] and its developments propose such hierarchies through catalogues that organize previously

accumulated design knowledge per type of NFR (e.g. a catalogue about security, another about performance). In particular, NFRF catalogues support the discovery of inconsistencies when there are qualities that require other qualities.

Heuristic - To find an inconsistency situation in a SRS:

1. search two word expressions from a catalogue of qualities (or violation of qualities), and one of these qualities requires the fulfillment of the other quality, and
2. check whether their descriptions are incompatible with each other.

Example. “Real-time dependency on Availability incompatible.” This example was explained at the beginning of Section III.

IV. JIGSAW PUZZLE GAME IN ACTION

The goal of the consultation meeting is to use the jigsaw puzzle game to analyze conflicts in the requirements presented, the ultimate goal being to handle these conflicts, e.g. resolve or defer the resolution. The participants in the meeting will be the requirements engineers and stakeholders chosen according to the requirements and conflicts under analysis. The requirements engineers are the meeting facilitators.

The consultation meetings should be organized according to the following guidelines. These guidelines follow the established practice in RE [33], but were modified according to specific goals, namely to promote cooperation and team building, to improve communication and co-responsibility, as well as to induce creativity. These guidelines were derived from, and tested in the evaluation sessions, and are as follows:

1. The participants are asked to perform group work to assemble the jigsaw puzzle. This is arranged to be easy and quick. The goal is to promote a game playing context, “break the initial ice”, and call the participants’ attention to focus on the same “object”, thus fostering cooperation and communication.
2. While assembling the jigsaw puzzle participants discover that there is a way the pieces fit but not perfectly. The fact that the pieces don’t fit means that the way the requirements are written contains conflicts, which do not allow such a system to be built.
3. Once the puzzle is assembled, the participants are asked to read the text in the pieces, and scan what could be the possible sources of conflict. The goal is to get the participants to scan visually for the conflicts, instead of the conflicts being readout: this is important to improve the sense that the conflicts are everyone’s problem and require cooperation and co-responsibility. Participants are encouraged to allow themselves to be questioned by the information the pieces convey, and to freely speak about any comments, and questions that came to their minds, as well as, to interact with other participants. Obviously, if the participants have difficulty in trying to identify the conflict cases, the facilitators can guide them.
4. Upon the discovery of a possible conflict, the group is challenged to analyze and discuss that conflict (including if it is really a conflict) and achieve a consensus on how to

handle it, for instance, additional; information that could resolve the conflict, rewording the requirements text (say, to remove ambiguity), or defer the resolution, etc. This promotes cooperation and team work among stakeholders.

V. EVALUATION

A. Goals

The goal of the evaluation was to test the following hypothesis, concerning the use of the jigsaw puzzle game during stakeholder consultation meetings:

- 1) *Hypothesis H1:* The use the jigsaw puzzle game promotes a relaxed environment and thus team cooperation and creativity.

The following two hypotheses act as second level hypotheses that help to contribute to validate H1:

- 2) *Hypothesis H2:* The effectiveness in communication and handling of requirements conflicts is increased, when compared with the same tasks performed with a text presentation.
- 3) *Hypothesis H3:* Team work and communication are fostered, improving commitment of stakeholders in co-authoring of requirements and co-responsibility in handling of conflict.

B. Research Methodology

The ideal research methodology would have been a series of case studies, since the context is expected to play a role in meetings between engineers and stakeholders [12]. As the access to real consultation meetings required by case studies was not possible, the experiments emulating “real-life” meetings were chosen as the best possible evaluation method. The evaluation was done through three such experiments, with a mixed philosophical stance: positivist and constructivist [12]. The experiments were confirmatory since they were used to test the hypotheses stated above. The experiments were also exploratory because they were used to understand the capabilities and problems of the proposed metaphor and method of working with it, leading to improvements.

C. Experiments Setup

The 1st experiment had 1 session, it was mainly exploratory and the 5 participants had RE background. The 2nd experiment consisted of 3 sessions and the 3rd experiment had 2 sessions. Every session was attended by a group of 3 to 4 participants, where at least one participant was an RE expert, while the others were non-software engineers, and at least one had no RE knowledge to emulate the role of external stakeholder. In the 3rd experiment the diversity in participants’ background was extended with the inclusion of 1 participant with management background in each of the sessions. The 2nd experiment introduced a control group, asking the participants to analyze 2 different systems: one with a jigsaw puzzle presentation, the other with textual documentation. In the 3rd experiment, the design of the information given to the control group was improved, i.e., the “same amount and type” of information was given for the system described in text as for the one presented through the jigsaw puzzle. This was done by introducing in the

text presentation phrases saying “Detected conflicts between requirement X and requirement Y”, exactly for the same situations that were signaled in the jigsaw puzzle with interlocking shapes. The SRS used were: the CMS [17] (the requirements Availability, Reliability, Real-time, and Accuracy); the Health-Watcher system (HW), which is a web-based system to manage health-related complaints [32] (the requirements Availability, Performance, Security, and Standards); and the CAS system, which is a customer relationship management application [1] (the requirements Availability, Security, Multi-channel access, and Accurate and Up-to-date information). A more detailed description of the experiments and their results can be found in [29].

D. Results

Concerning the hypothesis H1, it was observed that the fun side of assembling the jigsaw puzzle clearly contributed to a more informal and relaxed environment. Participants had fun and enjoyed the sessions. They cooperated as a team with open-mind and looked for ideas to solve the problems. Some tried to assemble the jigsaw puzzle pieces with the picture upside down, which indicates they had their minds open to do things in an unusual way.

Concerning the hypothesis H2, the experiments’ results demonstrate that the jigsaw puzzle game exhibits strong potential to increase the effectiveness in communicating and handling conflict in stakeholder meetings, when compared to a textual presentation. The most significant result sustaining this statement is that all participants (17), except one (who showed no preference), preferred to work with the jigsaw puzzle than with the textual presentation. In experiment 1 (where 5 of the participants were knowledgeable in RE and thus able to evaluate the comparison of the proposed approach against text) the participants concluded: “the jigsaw puzzle-based presentation is really good in helping identifying problems and conflicts”. In experiment 1 the participants were able to discover all four conflicts the puzzle intended to make explicit, for the CMS SRS. In experiment 1 there was no comparison with a textual version. In experiment 2 the participants discovered all three conflicts in the CMS puzzle, but none in the textual version of the CMS (which had no visual cues for conflicts). In experiment 3 the participants discovered three of the four conflicts in the CMS puzzle, but just one in the textual version of the CMS (with visual cues for conflicts). It is also significant that a number of participants were aware of and explained the jigsaw puzzle features that make them prefer to work with it. These features are as follows:

- The text presented in the jigsaw puzzle pieces is simpler and easier to read, there are keywords, smaller sentences, the letters are bigger, the “important” information is bullet-pointed and some is highlighted. Both the relevant topics and the relations between pieces are easier to spot. Some aspects are accentuated, enabling the users to think if, for instance, the numbers (shown in more than one piece for the same or related aspects) should match up;
- The jigsaw puzzle is colorful;

- Users can move the jigsaw puzzle pieces making it a more flexible presentation to work with;
- It promotes group work: everyone is focused at the same time on only one object;
- By presenting the requirements relationships through the interlocking shapes, users understand (visually) that requirements have impact on each other.

Concerning hypothesis H3, it was clearly observed that electrical engineers, and managers with no SE background, did not have any problem in understanding both the task to perform and the information that was conveyed with the jigsaw puzzle game: that there were problems to solve. Beginning the meeting with assembling a jigsaw puzzle also contributed to making everyone feel at the “same level”. The contributions, in terms of inconsistencies and ambiguities identified as well as possible modes of handling them, came from participants in spite of their backgrounds. The easiness in participate as well as to perceive the message that there were problems enabled participants to behave co-responsibly and to contribute to the requirements specification. This demonstrates co-responsibility and co-authoring.

E. Insights and Discussion

The way the puzzle pieces are laid down imposes/proposes an order to scan the possible conflicts. In all cases users tended to analyze the relation of the first piece they picked (usually the top left piece) with the ones that were adjacent. The question raised was: is this a good or bad effect? Participants’ opinion was that the way the puzzle pieces are disposed allows controlling the focus of the discussion. According to participants’ opinion, the structure provided by the puzzle, though on the one hand promotes analysis in a certain order, on the other it does not limit people to find imperfections among pieces that are not directly connected. This opinion was supported by the results collected, since participants reported conflicts among pieces that were neither directly connected nor had badly fitting interlocking shapes. The discovery of conflicts not represented is normal since the jigsaw puzzle aims to represent just a subset of conflicts. As indicated by the participants’ feedback, the use of different shapes (and geometrical structures) is an interesting area for further exploration.

As previously explained there is a connection between ambiguity and relations among requirements with unknown influence but with a potential to be an inconsistency. In such cases, the analysis of the two pieces to decide if there is an inconsistency promotes the clarification of the interpretation each person gives to the parts of the SRS in appreciation. This clarification of the interpretation(s) is what is needed to resolve the ambiguity. The 2nd and 3rd experiments provided insight that in such cases it might not be needed to have a visual representation for the ambiguity. It appears to be sufficient to provide a visualization for the associated inconsistency. There is ambiguity in the word ‘up-to-date’ in the Accuracy requirement as it does not describe what is meant by “up-to-date information”. In those experiments there was no visualization for ambiguity in the word ‘up-to-date’. However,

the participants discovered (due to the jigsaw puzzle shape metaphor) both the ambiguity and the associated inconsistency “Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement” (see Section III.B.1).

The number of conflicts well communicated and thus identified, as well as the suggestions for their handling can improve even more significantly, with an interactive digital support. Several participants suggested having the jigsaw puzzle game as a digital interactive set up. In itself just the fact that there is digital support (other than pure text) for analysis and rationale descriptions (e.g. comments, arrows relating requirements) can bring enormous benefits: these analyses and rationale are not lost (are recorded), and can easily be distributed. If the digital support also provides intelligent real-time interaction, enabling that the user changes, for instance, the text of a requirement and the interlocking shapes are redesigned according to a recalculation of possibility of conflicts, then we can say the (digitally supported) tool is enabling us to move towards minimization of conflicts. Furthermore, if this progress in work is automatically recorded, its effects are visually observable by the users involved and can be discussed “now”, instead of in the next meeting. This would represent a huge improvement in handling of requirements conflicts. One interesting possibility would be to have 3D jigsaw puzzles to enable visualization of a bigger number of relationships among requirements, and possibly with several levels of abstraction. Some participants wished to retain the physical pieces: they did not want to lose the possibility to touch the pieces physically, neither the initial challenge of putting the pieces together physically and realize how they are physically linked. These participants accepted to work with the digital jigsaw puzzle, but they would like to have also the physical form.

F. Threats to Validity

The evaluation used was qualitative and “perhaps more” constructivist than positivist [12]. We deemed applicable the following strategies [7] for improving validity of constructivist research: 1) clarify bias, 2) rich, detailed descriptions and 3) report discrepant information. Concerning the clarification of bias (1), the following situations may have introduced bias by the researchers: a) in the experiment preparation, when formatting the text in the pieces, researchers may have assumed implicitly one of the interpretations for the grammatical structure of a phrase that has syntactic ambiguity. This was mitigated by the use of heuristics described in Section III.A.2. In the future the use of ambiguity detection tools can help to format the requirements text with minimum bias; b) during the experiment, as the investigators also acted as facilitators, there was the possibility, that they conducted the participants to the solution; c) when reporting and analyzing the results. Concerning the report of the experiments’ results we sought to produce detailed descriptions (2) as well as to report discrepant information, should it be the case (3). Strategies 2) and 3) were aimed to reduce researchers’ bias (1c).

With regard to construct validity (positivist view) we already acknowledged that the best methodology to evaluate

this approach would have been a case study, but this was not possible. Anyhow the experiments performed as emulations of real meetings between stakeholders and engineers provided a reasonable approximation of the context of a real meeting, and it is plausible that results obtained would be confirmed in real meetings. We acknowledge that what was observed and described concerning the ability of the proposed approach to foster team work in groups having elements with different backgrounds, improving both cooperation, and co-responsibility, provides a conclusion with limitations. In fact as the evaluation was not done with real stakeholders the participants did not have a priori any reason to create an undesirable environment. In the 3rd experiment the quality of the real meeting emulation was improved through the introduction of participants with management background.

Concerning internal validity due to confounding factors (positivist view) [12], familiarity and learning were avoided using, for each session, two different systems. The system presented with the jigsaw puzzle was different from the one presented in text. Still concerning confounding factors, after the 2nd experiment, the hypothesis was raised that participants were not given the same information (treatment) while working with the jigsaw puzzle, compared to working with the textual documentation. The difference was that the information about conflicts was provided (as jigsaw puzzle cues) when the requirements were presented through the jigsaw puzzle, but was not provided in the textual documentation. This difference was addressed in the 3rd experiment, where conflict information was provided as part of the textual presentation as well. Another possible confounding factor detected when analysing the 2nd experiment, was the tiredness effect in the participants. In the 2nd experiment the 1st system was always presented with the jigsaw puzzle game, and the 2nd using the textual representation. The participants' preference for analyzing conflicts in requirements, with textual versus jigsaw puzzle game, could very well be biased in favor of the jigsaw puzzle, just because when participants worked with the text presentation (always in the second half of a two hour session) they were already too tired. To deal with this in the 3rd experiment, in one session the text description was utilized first and vice versa in the other session. One other possible confounding factor is that the examples worked might not be comparable, and in particular the HW system could be considered simpler than the CMS, i.e., having requirements easier to examine. It is difficult to ensure that the requirements documentation for two systems are "similar" in "handling difficulty".

VI. DISCUSSION AND FUTURE WORK

Our user evaluation provides two key insights with regards to the state-of-the-art in stakeholder consultation meetings:

Firstly, the user experiments offer insights into the relevance and richness of the jigsaw puzzle metaphor as an adequate communication means to discuss requirements with stakeholders. This communication mechanism is adequate and relevant because it makes the conflicts in requirements explicit through an easily understandable language. These

characteristics together with the gaming nature of this language improve co-ownership, co-responsibility, and creativity. This insight is strongly supported by the improved effectiveness in communication and handling of requirements conflicts, as well by the reactions of the users who easily engaged in team work with a co-responsibility and creative attitude.

Secondly, the search for such a communication mechanism, to make requirements conflicts explicit, brought into the conclusion that it is crucial to separate the processing of the information about the conflict from the issue of communicating that conflict. This separation is essential to permit a user-centered design of the communication of conflicts in RE.

Concerning future work, the possibility to have the jigsaw puzzle pieces as a digital interactive real-time mechanism was keenly advocated by the participants. We do believe it would bring enormous potential to the jigsaw puzzle game. However, we are equally mindful that the tactile, physical nature of the pieces was also highlighted a very positive aspect of the game. Any digital variant would need to preserve this tactile sensory experience. Certainly future work should include more experiments: with real stakeholders and software engineers in a system under development.

VII. CONCLUSION

We propose a jigsaw puzzle game and a method to effectively address the challenge of presenting the requirements and their conflicts to stakeholders, in a way that fosters co-responsibility and co-ownership regarding the conflicts and their resolution. The jigsaw puzzle metaphor is a novel contribution in the area of visual metaphors, and in particular as a communication mechanism to make conflicts explicit during stakeholder consultation meetings. The novelty of this metaphor resides in: 1) being an easy to understand language for users from non-computing backgrounds; 2) providing easy to understand analogies with well-understood concepts such as building a system out of pieces, and misshapen pieces representing that the system cannot be built with such faulty pieces; and 3) its gaming nature.

The evaluation performed demonstrates that the jigsaw puzzle game does increase effectiveness of communication and handling of conflicts in requirements, when compared with textual presentation of requirements. It fosters team work and communication, and improves commitment of stakeholders in co-authoring of requirements and co-responsibility in resolution of conflict. As the proposed solution builds upon a game (the jigsaw puzzle), it promotes a relaxed environment and thus creativity.

The separation of the processing of the information about the conflicts from the issue of communicating those conflicts is a core design principle of the jigsaw puzzle game. Such a separation is crucial to permit a user-centered design of communication of conflicts in RE and can act as a stepping stone for future research in this regard.

REFERENCES

- [1] D. Ayed, T. Gessler, Dynamic Variability in complex, Adaptive systems, Deliv. D6.1 DiVA EC project, 2009.

- [2] M. Balzer, A. Noack, O. Deussen, and C. Lewerentz, "Software landscapes: Visualizing the structure of large software systems", *Symposium on Visualization*, pp. 261–266, 2004.
- [3] D.M. Berry, E. Kamsties, and M.M. Krieger, From contract drafting to software specification: Linguistic sources of ambiguity, <https://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf>, November, 2003.
- [4] S. Boccuzzo, and H.C. Gall, "Cocoviz: Towards cognitive software visualization", *IEEE Int. Workshop on Visualizing Soft. for Understanding and Analysis*, 2007.
- [5] S. Bresciani, and M. Eppler, "The risks of visualization. A classification of disadvantages associated with graphic representations of information", *ICA Working Paper #1/2008*, University of Lugano (USI), 2008.
- [6] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishers, 2000.
- [7] J.W. Creswell, *Research design: Qualitative, quantitative and mixed methods approaches*, 2nd Edition, Sage, 2002.
- [8] L.M. Cysneiros, E. Yu, and J.C.S.P. Leite, "Cataloguing non functional requirements as softgoals networks", *Workshop on RE for Adaptable Architectures*, 11th IEEE Int. Conf. Requirements Engineering, pp. 13-20, 2003.
- [9] J.S. Davis, "Identification of errors in software requirements through use of automated requirements tools", *Information and Software Technology*, 31(9), pp. 472-476, 1989.
- [10] S. Diehl, *Software Visualization*, Springer, 2007.
- [11] S. Easterbrook, and B. Nuseibeh, "Using ViewPoints for inconsistency management", *Software Engineering Journal*, 11, pp. 31-43, BCS/IEE Press, 1996.
- [12] S.M. Easterbrook, J. Singer, M. Storey, and D. Damian, "Selecting empirical methods for software engineering research", in *Guide to Advanced Empirical Software Engineering*, F. Shull, and J. Singer, Eds. Springer, 2007.
- [13] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, "Viewpoints: A framework for integrating multiple perspectives in system development", *Int. Journal of Soft. Eng. and Knowledge Eng.*, 2(1), pp. 31-58, 1992.
- [14] O.C.Z. Gotel, F.T. Marchese, and S.J. Morris, "The potential synergy between information visualization and software engineering visualization", 12th International Conference on Information Visualization, 2008.
- [15] IEEE-SA Standards Board, *IEEE Recommended Practice for Soft. Req. Spec.*, IEEE Std. 830-1998, IEEE, 1998.
- [16] International Standard Organization, *ISO/IEC 25010:2011: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*, ISO, 2011.
- [17] J. Kienzle, N. Guelfi, and S. Mustafiz, Crisis Management Systems: A case study for Aspect-Oriented Modeling, <http://www.cs.mcgill.ca/~joerg/taosd/TAOSD/TAOSD.html>, May, 2009.
- [18] E. Knauss, K. Schneider, and K. Stapel, "A game for taking requirements engineering more seriously", 3rd Int. Workshop on Multimedia and Enjoyable Req. Eng., 2008.
- [19] J. Lee, J. Kuo, N. Hsueh, and Y. Fanjiang, "Trade-off requirement engineering", in *Soft. Eng. with Comp. Intelligence*, J. Lee, Ed. Springer, pp. 51-72, 2003.
- [20] N. Maiden, A. Gizikis, and S. Robertson, "Provoking creativity: imagine what your requirements could be like", *IEEE Software*, 21(5), pp. 68-75, 2004.
- [21] N. Maiden, and S. Robertson, "Integrating creativity into requirements processes: experiences with an air traffic management system", 13th IEEE Int. Conf. RE, 2005.
- [22] N. Maiden, C. Ncube, and S. Robertson, "Can requirements be creative? Experiences with an enhanced air space management system", *Proc. ICSE*, 2007.
- [23] B. Meyer, "On formalism in specifications", *IEEE Software*, 2(1), pp. 6-26, 1985.
- [24] L. Mich, M. Franch, and P.N. Inverardi, "Market research for requirements analysis using linguistic tools", *Requirements Engineering*, 9(1), pp. 40-56, 2004.
- [25] L. Mich, C. Anesi, and D.M. Berry, "Applying a pragmatics-based creativity-fostering technique to requirements elicitation", *Requirements Engineering*, 10(4), pp. 262-275, 2005.
- [26] N. Millard, P. Lynch, and K. Tracey, K.; "Child's play: using techniques developed to elicit requirements from children with adults", 3rd Int. Conf. RE, pp. 66-73, 1998.
- [27] J. Noppen, J., *Imperfect Information in Software Design Processes*, PhD Thesis, Enschede, The Netherlands, 2007.
- [28] B. Nuseibeh, *A Multiple Perspective-Framework for Method Integration*, PhD Thesis, Department of Computing, Imperial College, London, UK, 1994.
- [29] M. Pinto-Albuquerque, *Communicating Conflict and Ambiguity in Requirements Engineering*, PhD Thesis, Lancaster University, UK, 2013.
- [30] A. Sardinha, R. Chitchyan, N. Weston, P. Greenwood, and A. Rashid, "EA-Analyzer: automating conflict detection in aspect-oriented requirements", 24th IEEE/ACM Int. Conference on Automated Software Engineering, 2009.
- [31] R. Smith, O. Gotel, "Gameplay to introduce and reinforce requirements engineering practices", 16th IEEE Int. Conf. on Requirements Engineering, pp. 95-104, 2008.
- [32] S. Soares, P. Borba, and E. Laureano, "Distribution and persistence as aspects", *Software: Practice and Experience*, 36 (7):711–759, 2006.
- [33] I. Sommerville, and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, Wiley, 1997.
- [34] I. Sommerville, P. and Sawyer, "Viewpoints: principles, problems and a practical approach to requirements engineering", *Annals of Soft. Eng.*, 3, pp. 101-130, 1997.
- [35] B. Tversky, J.B. Morrison, and M. Betrancourt, "Animation: can it facilitate?", *International Journal of Human Computer Systems*, 57, pp. 247-262, 2002.
- [36] R. Wettel, and M. Lanza, "Program comprehension through software habitability", 15th Int. Conference on Program Comprehension, pp. 231 - 240, 2007.
- [37] H. Yang, A. de Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing anaphoric ambiguity in natural language requirements", *Requirements Engineering*, 16(3), pp. 163-189, 2011.

Automated Support for Combinational Creativity in Requirements Engineering

Tanmay Bhowmik*, Nan Niu†, Anas Mahmoud*, and Juha Savolainen‡

* Department of Computer Science and Engineering, Mississippi State University, USA

† Department of Electrical Engineering and Computing Systems, University of Cincinnati, USA

‡ Danfoss Power Electronics A/S, Graasten, Denmark

tb394@msstate.edu, nan.niu@uc.edu, amm560@msstate.edu, JuhaErik.Savolainen@danfoss.com

Abstract—Requirements engineering (RE), framed as a creative problem solving process, plays a key role in innovating more useful and novel requirements and improving a software system’s sustainability. Existing approaches, such as creativity workshops and feature mining from web services, facilitate creativity by exploring a search space of partial and complete possibilities of requirements. To further advance the literature, we support creativity from a combinational perspective, i.e., making unfamiliar connections between familiar possibilities of requirements. In particular, we propose a novel framework that extracts familiar ideas from the requirements and stakeholders’ comments using topic modeling and applies part-of-speech tagging to obtain unfamiliar idea combinations. We apply our framework on two large open source software systems and further report a human subject evaluation. The results show that our framework complements existing approaches by generating original and relevant requirements in an automated manner.

Index Terms—Requirements engineering; creativity; topic modeling; requirements elicitation

I. INTRODUCTION

Much of traditional requirements engineering (RE) has considered that requirements exist in the stakeholders’ minds in an implicit manner [1], and has focused on models and techniques to aid identification and documentation of such requirements. Modern software industry, however, has become extremely competitive as we find multiple software products striving to serve the users in the same application domain. In order to sustain, a software system needs to distinguish itself from other similar products and consistently enchant customers with novel and useful features. As a result, requirements engineers need to create innovative requirements in order to equip the software with competitive advantage. To that end, RE, framed as a creative problem solving process, plays a key role in innovating more useful and novel requirements, thereby improving a software system’s sustainability [2], [3].

Creativity, a multidisciplinary research field, can be considered as “the ability to produce work that is both novel (i.e., original and unexpected) and appropriate (i.e., useful and adaptive to task constraints)” [4]. According to Maiden *et al.* [2], creativity in RE is the capture of requirements that are new to the project stakeholders but may not be historically new to humankind. It has been suggested that stakeholders may obtain creative requirements by exploring, combining, and transforming existing ideas in the conceptual domain [2], [5]. Note that creativity may be more related to novelty, while

innovation also requires some demonstrated value or utility. In this sense, our current work focused more on creativity.

In order to aid creativity in RE, recent research has investigated several approaches. Maiden and colleagues [3], [6], [7], [8] conducted creativity workshops on exploring technical and psychological aspects of creativity and suggested integrating these aspects in the RE process. Techniques, such as generating requirements with scenario, have been proposed to support creativity while exploring information analogical to the current context [9], [10]. In a recent study, Hariri *et al.* [11] presented a framework to obtain requirements by mining feature descriptions of similar products from online product listings. These contemporary approaches facilitate creativity by *exploring* a search space of partial and complete possibilities of requirements. To further advance the literature, we support creativity from a *combinational* perspective, i.e., making unfamiliar connections between familiar possibilities of requirements [5], [12].

In this paper, we propose a novel framework that mines ideas familiar to the stakeholders and creates new requirements by obtaining unfamiliar connections. It has been suggested that people belonging to the same social group are generally interested in similar ideas and share common knowledge [13], [14]. Accordingly, in order to extract familiar ideas, we mine the requirements commonly discussed by distinct stakeholder groups. To that end, we first group the stakeholders by clustering the network created based on stakeholders’ social interaction. Then, we obtain ideas in terms of dominant topics [15] by applying Latent Dirichlet Allocation (LDA), the most commonly used technique for topic modeling in natural language processing [16]. We further achieve unfamiliar combinations of the dominant ideas by exploiting part-of-speech (POS) tagging [17]. We apply our framework on Firefox¹ and Mylyn [18], two large open source software (OSS) systems. We further conduct a human subject evaluation and the results indicate promising practical implications of our framework.

The contributions of our work lie in an advancement of the current solutions that facilitate creativity practice in RE. Our framework provides automated support for combinational creativity and complements existing approaches by generating original and relevant requirements. The rest of the paper is

¹<http://www.mozilla.org/en-US/firefox/new/>

organized as follows. Section II covers background information and related work. Section III introduces our framework. Section IV describes the creation of new requirements for our subject systems. Section V details the human subject evaluation followed by Section VI presenting further discussion and the limitations of the work. Finally, Section VII concludes the paper with an outline of our future work.

II. BACKGROUND AND RELATED WORK

A. Creativity in RE

Creative ideas: Being novel and being appropriate are the two intrinsic attributes of an idea to be creative [4]. An idea can be novel from three different aspects: H-Creativity — new to a person-kind (i.e., historically creative) [5], P-Creativity — new to a person but not to the person-kind or others (i.e., psychologically creative) [5], and S-Creativity — idea for a specific task which is novel in the particular situation or domain (also known as situated creativity) [19]. Meanwhile, an idea is appropriate if it is useful to accomplish a task and can be adapted following the task constraints [4]. According to Maher *et al.* [20], from a design perspective, an idea can be creative if it instigates surprise in terms of deviation in patterns of outcomes. Maiden and colleagues [2], however, suggest creativity in RE to be mostly situated creativity, i.e., creating requirements and other outcomes new to project stakeholders but need not be historically new.

Over the last decade, several techniques have been proposed in order to measure the novelty of a new requirement. Ritchie [21] posited a set of formal criteria that could be applied to assess the creative behavior of software programs. Measuring dissimilarity to existing domain examples could be a way of determining novelty of a requirement [12]. In order to invent requirements from software, Zachos and Maiden [10] exploited requirements similarity matching engines and judged novelty by computing dissimilarities among analogical matches. In the creativity framework proposed in this paper, we exploit the idea of measuring dissimilarity in finding unfamiliar idea combinations.

Categories of creativity: Following Boden [5], creativity in RE is categorized into three groups depending on the techniques and heuristics used [12]. 1) In *exploratory creativity*, creative requirements are obtained by exploring a partial and complete possibilities in the search space. This exploration is guided by rules and task constraints specific to the intended software system. 2) *Combinational creativity* is achieved by making unfamiliar connections between known requirements in a familiar setting. 3) The third way of accomplishing creativity in RE is to challenge the constraints on the search space and to enlarge the space of possible requirements to be explored. Creativity attained by this means is known as *transformational creativity*.

Figure 1 presents a conceptual picture of the three categories of creativity. Let us assume a creativity scenario for a hypothetical software product *S*: “provide access control” is a current requirement and limitation on available hardware is an initial constraint. Let XYZ be a search space with possible

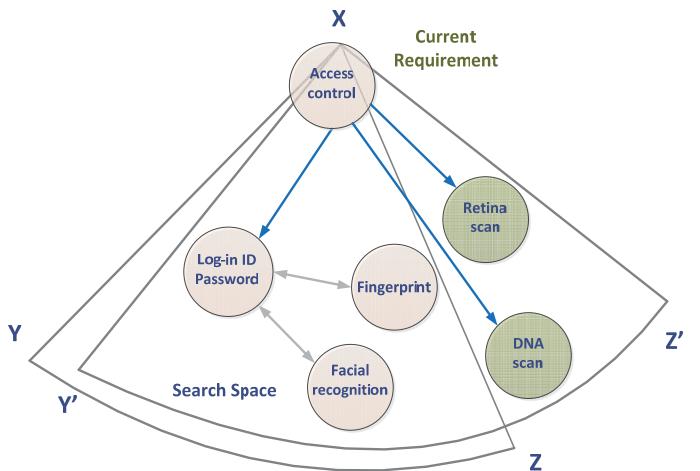


Fig. 1. Categories of creativity based on techniques and heuristics (adapted from [12]).

requirements “log-in ID and password”, “finger print”, and “facial recognition”. Provided that they satisfy the system constraints, using any of these options for access control is an instance of exploratory creativity. Combination of two apparently different access control means, such as log-in ID and password along with finger print, or log-in ID and password combined with facial recognition, can be considered as combinational creativity. Now, let us further consider that the initial constraint on hardware limitation is relaxed and we enlarge the search space towards the biometric direction, thereby obtaining the new search space $XY'Z'$. Options, such as DNA and retina scan, could also be available due to this expansion, an instance of transformational creativity.

On the way to creative requirements: Research has conducted creativity workshops and presented several frameworks in order to incorporate creativity techniques and heuristics in a direct or indirect manner. Zachos and Maiden [10] conducted creativity workshops in order to identify requirements for the Fiat real-time parking space booking system. They performed an analogical mapping between hotel reservation and parking space booking, and explored online hotel reservation systems to create requirements. In order to discover features for a future air space management software system, exploratory creativity was followed during the analysis of requirements and emergent system properties [3]. Lutz and colleagues [22] presented an approach that performed KAOS Obstacle Analysis to explore requirements in a space defined by obstacles for a safety-critical, autonomous system. Salinesi *et al.* [23] proposed a prototype tool that performed requirements-based product configurations within constraints. This tool discovered various permitted features for a new product in a product line. The *i*/TROPOS* proposed by Fuxman *et al.* [24] exploited model checking techniques on the explored space of specification properties in an attempt to avoid unreasonable requirements. All these frameworks and tools mostly incorporate exploratory creativity, directly or indirectly, and are concerned with creating requirements for a new system or a product line.

Lately, the collaborative nature of creativity in RE has been highlighted by Mahaux and colleagues [25]. Their research shows that people often need to work collaboratively to be creative and provides a framework characterizing the collaborative creative process in RE. Following their findings, in this paper, we consider the collaborative attribute of creativity and take into account not only the requirements descriptions but also the comments posted by stakeholders during their collaboration. To that end we utilize the concept of stakeholders' social network in finding their collaboration groups.

B. Stakeholders' Social Network in Software Engineering

Stakeholders and social networks based on their interactions have been widely studied in RE and other software engineering areas, e.g., software maintenance. Damian *et al.* [26] presented the concept of requirements-centric social network by defining social network among stakeholders working on same or inter-dependent requirements. Begel *et al.* suggested that people could “be friends” by working on the artifacts they share among them [27]. In order to create a visual representation for stakeholders' socio-technical relationship, Sarma *et al.* [28] considered both emails among developers and comments in Bugzilla issue tracking system. Posting and reading comments by stakeholders were also considered by Wolf and colleagues as a means to represent communication flow [29]. Building on the prior research, we consider in our work posting comments and artifacts on issue tracking systems as social interaction among stakeholders.

C. Topic Modeling with Latent Dirichlet Allocation (LDA)

LDA was first introduced by Blei *et al.* [16] as a statistical model for automatically discovering topics in large corpora of text documents. The main assumption is that documents in a collection are generated using a mixture of latent topics, where a topic is a dominant theme that describes the concept of the corpus's subject matter. LDA's scalability, language-independency, as well as its ability to work with incomplete text have made it an appealing analysis model for several software engineering activities [30], [31], [32]. Because the requirements of a software system as well as stakeholders' comments typically contain textual descriptions, LDA becomes particularly useful for our framework. Such textual content can be analyzed to produce latent topic structures for the requirements where every requirements description, associated with stakeholder comments, is analogous to an individual document.

Mathematically, a topic model can be described as a hierarchical Bayesian model that associates a document d in a document collection D with a probability distribution over a number of topics T . In particular, each document d in the collection ($d_i \in D$) is modeled as a finite mixture over T drawn from a Dirichlet distribution with parameter α , such that each d is associated with each ($t_i \in T$) by a probability distribution of θ_i . On the other hand, each topic t in the identified latent topics ($t_i \in T$) is modeled as a multidimensional probability distribution, drawn from a Dirichlet distribution β , over the set

of unique words in the corpus (W), where the likelihood of a word from the corpus ($w_i \in W$) to be assigned to a certain topic t is given by the parameter ϕ_i .

LDA takes the documents collection D , the number of topics K , and α and β as inputs. Each document in the corpus is represented as a bag of words $d = \langle w_1, w_2, \dots, w_n \rangle$. Since these words are observed data, Bayesian probability can be used to invert the generative model and automatically learn ϕ values for each topic t_i , and θ values for each document d_i . In particular, using algorithms such as Gibbs sampling [33], an LDA model can be extracted. This model contains, for each t , the matrix $\phi = \{\phi_1, \phi_2, \dots, \phi_n\}$, representing the distribution of t over the set of words $\langle w_1, w_2, \dots, w_n \rangle$, and for each document d , the matrix $\theta = \{\theta_1, \theta_2, \dots, \theta_n\}$, representing the distribution of d over the set of topics $\langle t_1, t_2, \dots, t_n \rangle$. The topic with the highest probability of occurrence in d is the most dominant topic for d . Therefore, for the document collection D , the topic that becomes dominant the greatest number of times is the most dominant topic for D .

Topic modeling in software engineering: Topic modeling has recently been used in several research areas of software engineering, such as mining software repositories (MSR) [15], [32], [34], requirements traceability [30], and software evolution [35]. Linstead *et al.* [15] applied LDA topic modeling technique on the source code of different versions in order to analyze software evolution. Linstead and colleagues [34] further used topic modeling on Internet-scale software repositories, and summarized program function and developer activities by extracting topic-word and author-topic distributions. The use of topic modeling over source code has been validated and it has been found that the evolution of source code topics is indeed caused by actual change activities in the code [32]. Asuncion *et al.* [30] proposed an automated technique that combined traceability with topic modeling and performed semantic categorization of artifacts during the software development process.

The above efforts follow a common approach in that they apply topic modeling on source code written in computer programming languages. In the creativity framework presented in this paper, one of the objectives is to extract existing ideas from documents mostly written in a natural language (e.g., English). To that end, we adopt LDA, perhaps the most proven topic modeling technique for NLP, thereby generating the underlying dominant themes from requirements and comments posted by stakeholder groups. In the next section, we present a detailed discussion of our creativity framework.

III. OUR CREATIVITY FRAMEWORK

Figure 2 presents an overview of our framework that applies a combinational creativity technique to obtain new requirements for an existing system. The framework starts with a social network based on stakeholders' social interaction, goes through several phases involving techniques, such as clustering, topic modeling, POS tagging, and similarity analysis, and ultimately creates new requirements in an automated manner. These requirements could be considered as an initial baseline

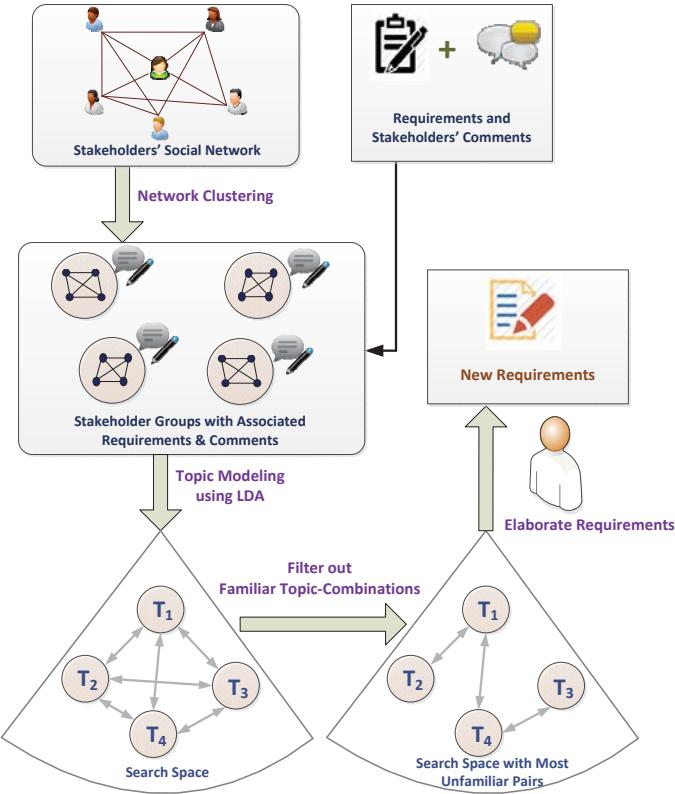


Fig. 2. A framework for combinational creativity.

for creative requirements that can further be discussed among stakeholders for analysis and modification. In the rest of this section, we discuss the phases of this framework in details.

1. Building the social network: The first phase of our framework is to build a weighted connected graph representing the stakeholders' social network. This network should be built based on stakeholders' communication, i.e., two people communicating among themselves should be connected by an edge and how strongly (or frequently) they communicate should be reflected by the edge weight. As several communication means could be followed by stakeholders (cf. Section II-B), our framework does not set any restriction on what activities should be considered as stakeholders' social communication. Depending on the practice followed in a specific software development environment, any set of well defined and properly recorded communication means should be suitable.

2. Clustering the social network: This phase involves clustering the social network in order to obtain stakeholders' social groups. The idea is that the members in the same group have more frequent interaction, whereas there is sparse communication among people belonging to different groups [14]. Our framework is flexible from the perspective of clustering in that any suitable network clustering algorithm [36] can be used as long as necessary information required for the clustering algorithm is available for the social network in concern. Tool supports, both commercial and open source, are available that can be used to perform this clustering activity [37], [38].

3. Extracting familiar ideas: As people belonging to the same social group are generally interested in similar ideas [13], [14], this phase of our framework involves identifying such ideas. In doing so, requirements and comments posted by each member in a social group are collected as text documents, one for each stakeholder. Let us assume that i number of groups have been obtained after the clustering phase. If there are N_i number of members in the i -th group, there will be a collection of N_i documents. LDA is applied on each document collection N_i in order to obtain the topic-word distribution matrix ϕ and document-topic distribution matrix θ . Irrespective of the size of the document collection, LDA always generates t topics where t is a positive natural number (often 100 [15]) chosen by the user. As both ϕ and θ provide the probability distribution of a large number of words and topics respectively, the number of words and topics should be considerably reduced to avoid an explosion of idea combinations in the later phases of this framework. To that end, the following procedure is pursued for each document collection N_i .

- We use the five most probable words from each topic as representatives of the topic's subject; 5 to 3 words were found to be sufficient to convey the topic's subject [39].
- We use the most probable (dominant) topic of each document to represent the document. Formally, a dominant topic can be described as $\theta_{i,j} = \max\{\theta_{h,j}, h = 1 \dots k\}$.
- The topics are sorted in descending order based on the number of documents they are dominating.
- These numbers are plotted against the topics and the cutoff is taken based on the trend, thereby obtaining a smaller number of topics for the social group.

4. Obtaining unfamiliar combinations of familiar ideas:

Extracted dominant topics provide us a search space of familiar ideas (cf. Figure 2). Our objective is to make unfamiliar connections between familiar possibilities in the search space. To that end, we aim to combine words from two topics, one word from each topic, coming from two different stakeholders' groups. If there are 10 groups with 5 dominant topics per group and 5 words per topic, there will be $10C_2 \times 25C_1 \times 25C_1 = 28,125$ unique word pairs. This will lead to a combinatorial explosion problem for systems with a large number of diverse stakeholders. In order to tackle this issue, we follow the work on semantic analysis in RE [40], [41] to tease out the action-oriented theme of a requirement. Such theme, according to Fillmore's case theory [42], can be characterized by the *verb* in a requirements description and the *direct object* that the verb acts on. Building upon this knowledge, we structure this phase of the framework with the following two steps, as illustrated in Fig. 3.

- *Flipping the part-of-speech:* For each topic word, we identify its common POS in the existing requirements and comments over the original corpus using a POS tagger. POS tagging is recently being used in text based software engineering tools, such as SWUM [43] and POSSE [44]. We take the most common verb from a topic and the most common noun (object) from another, where two

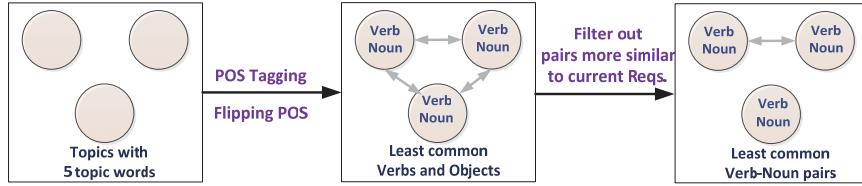


Fig. 3. Finding the least common verb-noun pairs: the leftmost box in the picture contains three topics, represented by three circles, with 5 words each.

topics belong to two separate groups of stakeholders, and consider the words as noun (object) and verb respectively. We identify all such verb-noun pairs.

- **Finding system specific unfamiliar pairs:** To further ensure unfamiliarity, we rank the verb-noun pairs based on their average textual similarities [45] with the current requirements. Then, we filter out the combinations with higher similarity values following a relative filtering approach [45], thereby reducing the search space to most unfamiliar verb-noun pairs (cf. Fig. 2).

5. Elaborating requirements from verb-noun pairs: All the word pairs obtained from the previous phase are presented to a human analyst, preferably a stakeholder proficient in the software's functional attributes. The analyst is also supplied with the semantic and contextual information about the words, as well as more frequently used phrases around the words in the existing requirements. Equipped with these resources, the analyst shall phrase statements following some syntax, e.g., subject + verb + noun/object, and further elaborate the statement using contextual information to obtain new requirements. The subject could be the software system itself (e.g., Firefox) or a suitable phrase from the context. Our framework is flexible in this regard as the analyst is free to elaborate the requirements in her own words as long as the *ideas* provided by the verb-noun pairs are reserved. Further technical details of the framework is presented in Section IV.

IV. CREATING REQUIREMENTS USING OUR FRAMEWORK

This section explains our procedure of examining how the proposed framework supports combinational creativity in RE. In particular, we detail the activities we perform to tease out original, unexpected, useful, and adaptive requirements following the specific phases discussed in Section III.

A. Methodology

In order to test our framework, we select two OSS systems: Firefox and Mylyn [18]. We select these projects as our subject systems for a number of reasons. First, they are large OSS systems and were previously studied in software engineering research [18], [46]. Second, they are very successful applications and can be considered representatives of their own domains. Third, the relevant data about these systems, required to conduct this study, are freely available online over Bugzilla. This enables other researchers to replicate our study. Next is a brief description of our chosen systems.

- **Firefox:** A very successful open source project and a dominating Web browser since its first release in 2004.

From November 2004 to June 2011, Mozilla released Firefox stable versions 1.0 through 5.0 and after that made some rapid releases.² We collect data about the closed requirements (feature requests) of the stable versions.

- **Mylyn:** A stable plug-in that monitors programmer activity in the Eclipse IDE [18]. It was first started as a part of the PhD thesis supervised by Gail Murphy at the Software Practices Lab at UBC.³ We consider the closed requirements of Mylyn from its starting in 2005 till February 2012.

For every requirement, we collect information as follows: requirement ID, description, comments, proposer (i.e., stakeholder who proposed the requirement), and stakeholders posting comments and artifacts. All the information, directly available from the requirements page, is collected by running a Web scraping tool written in Java. Table I presents the collected data that we analyze for the subject systems. Note that we observe many requirements marked as duplicates (specially in case of Firefox), and exclude them from our study.

B. Creative Requirements via Idea Combinations

Building the social network: For each subject system, the social network is a weighted graph where each node represents a stakeholder. An edge in the graph represents the communication among two stakeholders and the weight of this edge indicates the total instance of communications between them. To define the weighted edges, we adopt the approach presented by Wolf *et al.* [29]. Let X and Y be two stakeholders and R be a requirement that both X and Y contribute to. We identify an edge XY representing communication between X and Y if: 1) X is the proposer of R or has posted a comment or artifact about R that is read by Y ; or 2) Y is the proposer of R or has posted a comment or artifact about R that is read by X . As issue trackers do not keep direct trace of a stakeholder's reading activity, following Wolf *et al.* [29], we assume Y read the information posted by X about R if and only if Y also made a posting. We aggregate such communication instances between X and Y over the analyzed history and obtain the weight of an edge XY .

Obtaining stakeholders' groups: In order to identify stakeholders' groups, i.e., people who interact more frequently among themselves, we cluster the social networks built in the previous phase. To that end, we use Ucinet [37],

²<http://www.mozilla.org/en-US/firefox/releases/>

³<http://www.eclipse.org/mylyn/about/>

TABLE I
DATA COLLECTION OF SUBJECT SYSTEMS

| System | Application domain | Analyzed history | # of req.s | Avg. # of comments per req. | # of code files | Written in |
|---------|--------------------|------------------|------------|-----------------------------|-----------------|-------------------|
| Firefox | Web browser | 2004–2011 | 983 | 18 | 1,968 (C/C++) | C/C++, JavaScript |
| Mylyn | Eclipse plug-in | 2005–2012 | 445 | 11 | 2,321 | Java |

TABLE II
CLUSTERING RESULTS

| System | # of stakeholders in social network | # of clusters (groups) | Avg. group size* | Fit value |
|---------|-------------------------------------|------------------------|-------------------|-----------|
| Firefox | 783 | 36 | 22 (± 8.29) | 0.783 |
| Mylyn | 136 | 9 | 16 (± 6.82) | 0.817 |

*The average value rounded to the next round number.

TABLE III
TOPICS OBTAINED

| System | # of clusters | Total # of topics | Avg. # of topics per cluster* | Possible unique word pairs |
|---------|---------------|-------------------|-------------------------------|----------------------------|
| Firefox | 36 | 318 | 9 (± 4.15) | 1,239,150 |
| Mylyn | 9 | 48 | 6 (± 3.72) | 29,864 |

*The average value rounded to the next round number.

which provides social network clustering and related features. Ucinet [37] takes the total number of expected clusters k as input and applies hierarchical clustering algorithm based on node similarities. In our context, a higher edge weight means higher similarity between nodes. The output is a text file that elicits the clusters and also provides a fit value where a lower fit value indicates better cluster quality [37]. For both Firefox and Mylyn, we start the clustering process with $k=2$, observe the fit values by gradually increasing k , and stop further clustering when there is no more reasonable decrease in the fit value. Table II presents the clustering results.

Familiar ideas from stakeholders' groups: Phase 3 of our framework applies LDA [16] on the requirements and comments from all the stakeholders in a social group. For this activity, we use JGibbLDA.⁴ This particular implementation uses Gibbs sampling for parameter estimation and inference [47]. From the topic-word matrix and document-topic matrix produced by JGibbLDA, we extract the dominant topics following the heuristics presented in Section III. Along with these matrices, JGibbLDA also produces a topic-words file from which we pick the top 5 words for each topic as topic words. It should be noted that we filter out common key words, such as Firefox and Mylyn, based on the system's context along with frequently used English stop words to avoid noise. However, we find some words appearing in multiple topics, such as Web in case of Firefox and task in case of Mylyn. In such cases, the word is assigned to the topic where it shows the highest probability of occurrence. The results after this phase is summarized in Table III.

The column 'Possible unique word pairs' in Table III presents the number of unique word pairs considering one word per topic from two different stakeholder groups. The high number of possible combinations makes it apparent that without further filtering, elaborating requirements from the word pairs will be very daunting for an analyst. Furthermore, not all word pairs will make much sense so that a meaningful

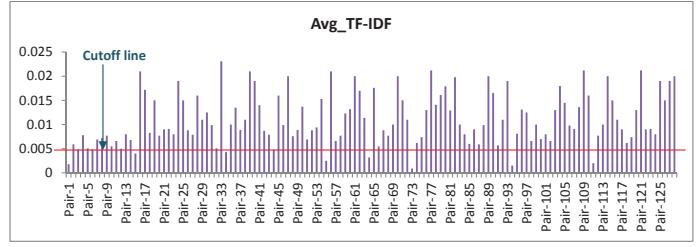


Fig. 4. Finding the least common verb-noun pairs for Mylyn.

requirement could be generated. The filtering phase that we go through next is specifically designed to tackle this issue.

Unfamiliar idea combinations for Firefox and Mylyn: This phase first uses Brill's tagger [17] to identify the most common POS for every topic word in the existing requirement descriptions over the original corpus. This allows us to find the most common noun (object) and the most common verb based on the word probabilities in the topic-words file produced in the previous phase. We consider them as least common verb and least common noun respectively, thereby producing the least familiar verb-noun pairs (from the system's perspective) for both Firefox and Mylyn.

Next, following the experience presented in our previous work [48], we calculate TF-IDF cosine similarities between a verb-noun pair and the existing requirements. We average the similarity values obtained for every pair and operationalize a relative filtering scheme to filter out the pairs with higher average similarities. To that end, we extract the verb-noun pairs with $average\ similarity \leq 0.20 * highest\ similarity$ and consider them as the final set of verb-noun pairs. Noted that the level of this cutoff is subject to calibrate for the requirements engineer to work with a manageable set of idea pairs for further expansion. Figure 4 demonstrates the filtering scheme for Mylyn. Table IV summarizes the results after this phase.

Elaborating requirements from pairs: In order to carry out this phase, we recruited a professional software engineer named Bob (pseudonym), working at a local software devel-

⁴<http://jgibblda.sourceforge.net/>

TABLE IV
UNFAMILIAR IDEA COMBINATIONS

| System | Possible idea combinations | | |
|---------|----------------------------|-------------------|-------|
| | Initial | After POS tagging | Final |
| Firefox | 1,239,150 | 2436 | 34 |
| Mylyn | 29,864 | 128 | 9 |

opment company. As part of his current job, Bob performs requirements analysis related activities at a regular basis. At work, he executes most of his development activities in Java using Eclipse IDE, and has been using both Firefox and Mylyn for several years. We provided Bob with the final set of word pairs (i.e., unfamiliar idea combinations), automatically generated contextual information for each word, and a set of sample templates to guide the elaboration. We asked Bob to come up with as many new requirements as possible using the word pairs within a time frame of two hours. Bob was requested to preserve the syntax ‘verb and noun (object) the verb acts upon’ as much as possible, and was allowed to use the Internet, if necessary. A researcher was present to explain the task and the provided artifacts, observe Bob’s activities during this elaboration process, and conduct an exit interview.

Figure 5 demonstrates the elaboration of a new requirement for Firefox. The words ‘text’ and ‘number’ were obtained from the topics *<support, text, lock, login, enter>* and *<build, compile, number, tool, item>* respectively. The dotted boxes contain recommended context information. Note that Bob did not use any recommended word from the topic list for ‘text’ while writing the requirement “Firefox user can text phone number from the web page.” After two hours, Bob elaborated 8 requirements for Firefox and 5 for Mylyn, as shown in Table V. The last column contains the final requirements descriptions after *refinement* and some nouns and verbs do not preserve their initially assigned POS anymore. Section VI provides further discussion on this issue. Next we present a human subject evaluation of our framework.

V. HUMAN SUBJECT EVALUATION

A. Study Setup

We recruited 29 developers with experience in Java and C#, including both undergraduate and graduate students and staff programmers from our institute. The developers participated voluntarily by responding to an email invitation. We made a confidentiality agreement with the participants to respect their anonymity. Each participant worked individually in a lab and began by signing the consent form. The demographic

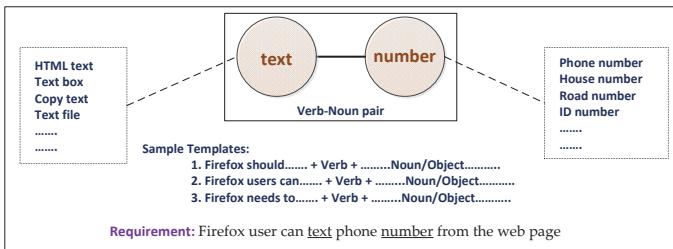


Fig. 5. Example for requirements elaboration.

TABLE V
ELABORATED REQUIREMENTS FOR FIREFOX AND MYLYN

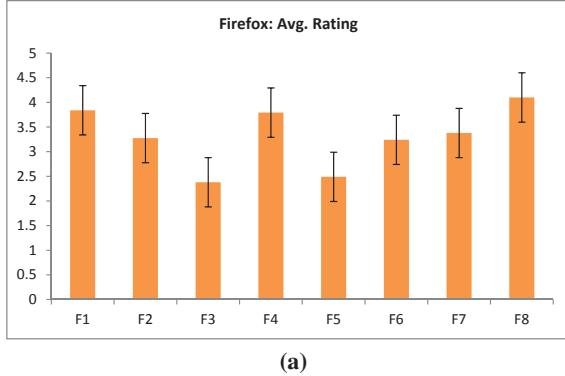
| System | Verb | Noun | Elaborated requirement |
|---------|------------|---------|---|
| Firefox | float | view | F1: Firefox should provide floating tab-view |
| | button | save | F2: Firefox should provide button for saving all tabs |
| | inform | count | F3: Firefox should inform user the count of open tabs |
| | select | save | F4: Firefox user can select text saving directly into files |
| | arrow | browse | F5: Firefox should provide arrow to browse tabs |
| | zoom | drag | F6: Firefox users can zoom in/out by a dragging slider |
| | text | number | F7: Firefox user can text phone number from the web page |
| | parallel | view | F8: Firefox can parallel tab-view |
| Mylyn | person | set | M1: Mylyn should provide options to personalize settings |
| | window | manage | M2: Mylyn should provide window for query manager |
| | credential | issue | M3: Mylyn should be credentialing issue tracking system |
| | plug | comment | M4: Mylyn should plug comment to issue tracking system |
| | shortcut | track | M5: Mylyn should provide shortcut to issue tracker |

information was also collected at this stage through a pre-study survey. The information included software development experience, familiarity with the subject systems, and the primary and secondary programming languages. The recruits reported a median of 3.5 years of software development experience. All the participants have had experience with Firefox (27 users only and 2 contributors) and 9 had knowledge about Mylyn. Irrespective of experience, a tutorial on the latest versions of Firefox and Mylyn was presented. Then the participant was given hard copies of the requirements along with the word pairs (cf. Table V) and was free to use the Internet for further information, if needed.

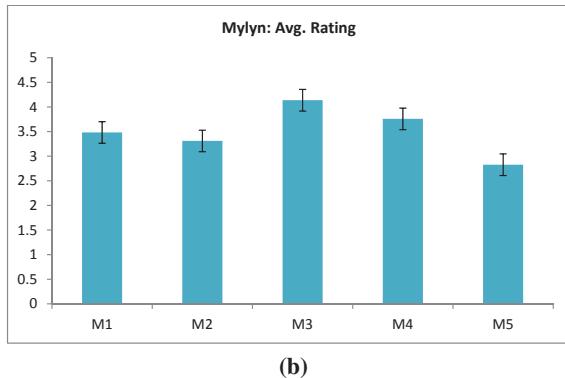
The participant’s task was to rate how creative each requirement was by using a 5-point Likert scale: 1=least innovative, 2=not innovative, 3=neutral, 4=innovative, 5=most innovative. It was explained that being innovative in this context meant: 1) Novel and new, as well as, 2) Relevant and useful for the intended software product. The participant was given an option to modify the requirement preserving the given term pair in case she felt some requirement to be less innovative. We asked every participant to work individually on all the requirements. A researcher was present to explain the study, to encourage the participant to think aloud during her session, to take notes, and to conduct an informal exit interview to elicit feedback about her experience. Each participant spent approximately 1.5 hours analyzing all the requirements and was presented with a \$10 gift card at the end as a token of our appreciation.

B. Results and Analysis

Figure 6 plots the average ratings reflecting how creative the participants perceive the requirements to be. The average ratings vary from being not innovative (e.g., F3 and M5) to innovative (e.g., F8 and M3) based on the 5-point Likert scale (cf. Section V-B). Overall, however, 6 Firefox and 4 Mylyn requirements can be considered innovative based on the average ratings. In order to assess the agreement among the participants on their ratings, we adopt *kappa statistic* (κ), a widely used measure of inter-rater reliability [45]. Kappa statistic returns a value in $[0, 1]$, where $\kappa=0$ shows no agreement and $\kappa=1$ suggests complete agreement. We find the average κ for Firefox and Mylyn requirements to be 0.79 and 0.67 respectively. According to the magnitude guideline provided by Manning *et al.* [45], these values indicate substantial



(a)



(b)

Fig. 6. Average ratings for the created requirements.

agreement among the participants. These results suggest that *our framework helps generate innovative requirements in an automated manner*. Some individual κ values, however, indicate slight agreement about the innovative values. Section VI provides further insight about this observation.

VI. DISCUSSION

So far, we have discussed how our combinational creativity framework can be applied to create new requirements for Firefox and Mylyn and our assessment on the innovative values of these requirements. In this section, we shed light on some observations and some lessons learned.

A. On Requirements Elaboration from Word Pairs

Compared to the possible idea combinations initially found (1,239,150 for Firefox and 29,864 for Mylyn), our framework has come up with a substantially smaller number of unfamiliar idea-pairs for further elaboration (cf. Table IV). We consider it to be the greatest strength of this framework as requirements elaboration is largely human intensive and a high number of recommended possibilities will make the elaboration phase tedious and overwhelming. Such a laborious activity in a framework like ours will break its main objective of offering automated support. However, the adjustable calibrations during the filtering phase always provide rooms to obtain higher number of unfamiliar idea combinations, if necessary.

One issue though, about our framework, pointed out by Bob (cf Section IV), is sticking to the idea of verb and noun (object) the way they are recommended. We observe Bob spending

quite some time to come up with requirements descriptions that preserve the given POS of the words. In several cases, although he started with the verb-noun outline, he rephrased the requirements (e.g., *F2* and *M5*) to make better senses out of them. When asked about his feedback on the elaboration process during the exit interview, one comment from Bob substantiates our observation.

I think the verb-object concept was helpful in the beginning to get some initial idea about a requirement. But strictly following that.... I don't think it will always work, nor I think it is necessary. In fact, sometimes it feels really over killing.

We would like to emphasize at this point that preserving the given POS should be considered as an initial guideline and may not be followed as a strict rule. Thereby our framework should provide enough space to phrase the requirements in a less restrictive manner.

B. On Innovative Requirements

Low average ratings: We observe some requirements received relatively low average ratings during the human subject evaluation. For example, *F3* and *F5* both received an average rating below 2.5 and based on the corresponding κ statistics (0.83 and 0.65), there was a strong sense of agreements among the participants. In case of *F3* (Firefox should inform user the count of open tabs), we heard a common argument regarding the usefulness of the requirement to the general users. One participant recommended "Instead, the count of tabs could be provided only if there are many tabs opened at once". For *F5*, most of the participants regarded it to be a redundant one. One participant recommended to make it an optional feature as she commented,

This requirement would not be very important to a user with a mouse that can easily click on each tab. However, it could be an optional one that is enabled if the user is using a touch screen device.

Requirements with higher ratings for innovation: Some requirements enjoyed higher average ratings with almost perfect agreements [45], e.g., in case of *F8* (Firefox can parallel tab-view), the average rating was 4.1 ($\kappa=0.81$). Some participants, even after rating it 'most innovative', were inspired to brainstorm around the parallel tab-view idea and enriched the requirement with further creative thoughts. The following statements from a participant (rated *F8* as 5) demonstrates this creativity instigating aspect of our framework.

It can be made as a button that will split the view of the window and will allow the user to drag tabs into each of them. Also a tab change shortcut can be applied to whichever window is being hovered over, allowing the user to retain the same full tab inventory.

Higher average ratings with limited agreements: In case of a couple of requirements, we observed relatively high creativity ratings but with slight agreements [45] among

the evaluators. *M4* is such an example ($\kappa=0.19$) where two participants identified it to be a particularly useful feature as they perceived it would reduce overhead due to context switches. One participant commented, “As I often contribute to a couple of open source projects, I think this feature will be useful for a user in posting comments and artifacts directly from the IDE”. However, many participants identified it as an over killing feature mentioning that, “All the issue tracking systems already provide facilities to post comments..... Don’t see any point of implementing it here”.

The overall findings attest the creative ability of our framework. In addition, the qualitative results obtained from the evaluation study indicate that the new requirements can also act as starting points and ignite improved creative thinking among analysts. This advances the innovative attributes in the requirements even further, thereby reinforcing the practicality of our framework.

C. Limitations

The work presented in this paper contains the development and demonstration of a conceptual framework, as well as a human subject evaluation. We discuss the limitations from both the framework and the evaluation related aspects.

From the framework perspective: Our framework is limited to its dependency on a large number of existing requirements preferably contributed by a diverse groups of stakeholders. The framework, as it is currently outlined, may not be applicable for a completely new software system in an emerging application domain. Furthermore, applying this framework to a system still at an infant stage may result in fairly limited outcomes. Our framework also largely depends on creating stakeholders’ social network that presents a reliable projection of their social interaction. As there exist several social network building techniques (cf. Section II-B), choosing the right means may be tricky. Our framework does not apply any restriction on how the social network should be built and we expect no further limitation along this line. Similar reasoning is also applicable for the network clustering techniques. Clustering the requirements and comments directly (e.g., [49]), instead of the social network, could be an alternative. We believe, however, considering the social network better reflects the collaborative nature of RE [25].

The limitations of topic modeling and POS tagging, such as working with a properly refined and grammatically well written text corpus [16], [17], are also relevant to our framework. However, the requirements and comments the framework tends to analyze are expected to be described in reasonably parsable statements. We also filter out unnecessary and most frequently used words before applying topic modeling. Therefore, we do not expect additional limitations due to these techniques. The practical implementation of our framework revealed a small number of requirements for the subject systems. This is mainly due to time constraints for the analyst and a more conservative filtering criterion (cf. Section IV-B). Further relaxed approach during these activities should help identify a higher number of new requirements.

From the evaluation perspective: An important threat to internal validity is related to the skills of the analyst who formulated the final requirements. It is therefore unknown how different RE skills can impact the evaluation results. We measure how innovative a new requirement is by taking an average of ratings at a 5-point Likert scale and also use the kappa statistics that show substantial agreement among the ratings. However, the participant’s level of familiarity with the subject system (specially in case of Mylyn [18]) might have a potential bias in the overall ratings. We mitigate this issue by presenting a tutorial about the subject systems and allowing access to the internet and discussion with the researcher conducting the study for further clarification. We have noticed frequent use of these resources by the participants, thereby improving the reliability of our findings.

VII. CONCLUSION

In this paper, we have contributed a novel framework that provides automated support for innovating requirements from a combinational creativity perspective [2], [5]. We have also presented a human subject study evaluating how effectively our creativity framework contributes novel and appropriate requirements for an intended software system. The results show that our framework successfully generates creative (i.e., original and relevant) requirements in an automated manner. Furthermore, the new requirements provoke creative thinking of an analyst, thereby improving the innovative aspects.

In future, we plan to reduce our framework’s dependency on existing requirements in order to expand its applicability to new software systems and live projects. We are also interested in using methods such as requirements template to facilitate the last phase of our framework, i.e., elaborating and refining requirements, thereby further increasing the automation degree. Finally, we intend to push our framework towards the dimension of transformational creativity in RE [12].

ACKNOWLEDGMENT

We express our sincere gratitude to all the participants for their valuable contributions to the study. This research is partially supported by the U.S. NSF (National Science Foundation) Grant CCF-1238336.

REFERENCES

- [1] J. Lemos, C. Alves, L. Duboc, and G. N. Rodrigues, “A systematic mapping study on creativity in requirements engineering,” in *Proceedings of the Annual ACM Symposium on Applied Computing (SAC)*, 2012, pp. 1083–1088.
- [2] N. Maiden, S. Jones, K. Karlsen, R. Neill, K. Zachos, and A. Milne, “Requirements engineering as creative problem solving: A research agenda for idea finding,” in *Proceedings of the International Requirements Engineering Conference (RE)*, 2010, pp. 57–66.
- [3] N. Maiden, C. Ncube, and S. Robertson, “Can requirements be creative? Experiences with an enhanced air space management system,” in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2007, pp. 632–641.
- [4] R. J. Sternberg, *Handbook of creativity*. Cambridge University Press, 1999.
- [5] M. A. Boden, *The creative mind: Myths and mechanisms*. Routledge, 2003.

- [6] N. Maiden, A. Gzikis, and S. Robertson, "Provoking creativity: Imagine what your requirements could be like," *IEEE Software*, vol. 21, no. 5, pp. 68–75, 2004.
- [7] N. Maiden, S. Manning, S. Robertson, and J. Greenwood, "Integrating creativity workshops into structured requirements processes," in *Proceedings of the ACM Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, 2004, pp. 113–122.
- [8] N. Maiden and S. Robertson, "Integrating creativity into requirements processes: Experiences with an air traffic management system," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2005, pp. 105–114.
- [9] I. K. Karlsen, N. Maiden, and A. Kerne, "Inventing requirements with creativity support tools," in *Requirements Engineering: Foundation for Software Quality*. Springer, 2009, pp. 162–174.
- [10] K. Zachos and N. Maiden, "Inventing requirements from software: An empirical investigation with web services," in *Proceedings of the Int'l Requirements Engineering Conference (RE)*, 2008, pp. 145–154.
- [11] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher, "Supporting domain analysis through mining and recommending features from online product listings," *IEEE Transactions on Software Engineering*, vol. 39, no. 12, pp. 1736–1752, 2013.
- [12] N. Maiden, "Requirements engineering as information search and idea discovery (keynote)," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2013, pp. 1–1.
- [13] R. S. Burt, "Structural holes and good ideas," *American Journal of Sociology*, vol. 110, no. 2, pp. 349–399, 2004.
- [14] P. Pirolli, "An elementary social information foraging model," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2009, pp. 605–614.
- [15] E. Linstead, C. Lopes, and P. Baldi, "An application of latent Dirichlet allocation to analyzing software evolution," in *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, 2008, pp. 813–818.
- [16] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.
- [17] E. Brill, "A simple rule-based part of speech tagger," in *Proceedings of the Workshop on Speech and Natural Language*, 1992, pp. 112–116.
- [18] M. Kersten and G. Murphy, "Mylar: A degree-of-interest model for IDEs," in *Proceedings of the International Conference on Aspect-Oriented Software Development (AOSD)*, 2005, pp. 159–168.
- [19] M. Suwa, J. Gero, and T. Purcell, "Unexpected discoveries and S-invention of design requirements: Important vehicles for a design process," *Design Studies*, vol. 21, no. 6, pp. 539–567, 2000.
- [20] M. L. Maher, K. Brady, and D. H. Fisher, "Computational models of surprise in evaluating creative design," in *Proceedings of the International Conference on Computational Creativity (ICCC)*, 2013, pp. 147–151.
- [21] G. Ritchie, "Assessing creativity," in *Proceedings of the AISB-01 Symposium on AI and Creativity in Arts and Science*, 2001, pp. 3–11.
- [22] R. Lutz, A. Patterson-Hine, S. Nelson, C. R. Frost, D. Tal, and R. Harris, "Using obstacle analysis to identify contingency requirements on an unpiloted aerial vehicle," *Requirements Engineering*, vol. 12, no. 1, pp. 41–54, 2007.
- [23] C. Salinesi, R. Mazo, D. Diaz, and O. Djebbi, "Using integer constraint solving in reuse based requirements engineering," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2010, pp. 243–251.
- [24] A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso, "Model checking early requirements specifications in tropos," in *Proceedings of the Int'l Symposium on Requirements Engineering (RE)*, 2001, pp. 174–181.
- [25] M. Mahaux, L. Nguyen, O. Gotel, L. Mich, A. Mavin, and K. Schmid, "Collaborative creativity in requirements engineering: Analysis and practical advice," in *Proceedings of the International Conference on Research Challenges in Information Science (RCIS)*, 2013, pp. 1–10.
- [26] D. Damian, S. Marczał, and I. Kwan, "Collaboration patterns and the impact of distance on awareness in requirements-centred social networks," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2007, pp. 59–68.
- [27] A. Begel, Y. Khoo, and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2010, pp. 125–134.
- [28] A. Sarma, L. Maccherone, P. Wagstrom, and J. Herbsleb, "Tesseract: Interactive visual exploration of socio-technical relationships in software development," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2009, pp. 23–33.
- [29] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2009, pp. 1–11.
- [30] H. Asuncion, A. Asuncion, and R. Taylor, "Software traceability with topic modeling," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2010, pp. 95–104.
- [31] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining concepts from code with probabilistic topic models," in *Proceedings of the International Conference on Automated Software Engineering (ASE)*, 2007, pp. 461–464.
- [32] S. Thomas, B. Adams, A. Hassan, and D. Blostein, "Validating the use of topic models for software evolution," in *Proceedings of the IEEE Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2010, pp. 55–64.
- [33] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed Gibbs sampling for Latent Dirichlet Allocation," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 569–577.
- [34] E. Linstead, S. Bajracharya, T. Ngo, P. Rigor, C. Lopes, and P. Baldi, "Sourcerer: Mining and searching Internet-scale software repositories," *Data Mining and Knowledge Discovery*, vol. 18, no. 2, pp. 300–336, 2009.
- [35] E. Linstead, P. Rigor, S. K. Bajracharya, C. V. Lopes, and P. Baldi, "Mining Internet-scale software repositories," in *Proceedings of the Neural Information Processing Systems (NIPS)*, 2007.
- [36] J. Wu, *Advances in K-means Clustering: A Data Mining Thinking*. Springer, 2012.
- [37] S. P. Borgatti, M. G. Everett, and L. C. Freeman, *Ucinet for Windows: Software for social network analysis*. Analytic Technologies, 2002.
- [38] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2009, pp. 361–362.
- [39] J. Chang, J. L. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *Proceedings of the Neural Information Processing Systems (NIPS)*, vol. 22, 2009, pp. 288–296.
- [40] S. Liaskos, A. Lapouchian, Y. Yu, E. Yu, and J. Mylopoulos, "On goal-based variability acquisition and analysis," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2006, pp. 79–88.
- [41] N. Niu and S. Easterbrook, "Extracting and modeling product line functional requirements," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2008, pp. 155–164.
- [42] C. Fillmore, "The case for case," in *Universals in Linguistic Theory*, E. Bach and R. Harms, Eds. New York: Holt, Rinehart and Winston, 1968, pp. 1–88.
- [43] E. Hill, *Integrating natural language and program structure information to improve software search and exploration*. PhD. Thesis, University of Delaware, 2010.
- [44] S. Gupta, S. Malik, L. Pollock, and K. Vijay-Shanker, "Part-of-speech tagging of program identifiers for improved text-based software engineering tools," in *Proceedings of the International Conference on Program Comprehension (ICPC)*, 2013, pp. 3–12.
- [45] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1.
- [46] S. Zaman, B. Adams, and A. E. Hassan, "Security versus performance bugs: A case study on firefox," in *Proceedings of the Working Conference on Mining Software Repositories (MSR)*, 2011, pp. 93–102.
- [47] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National academy of Sciences of the United States of America*, pp. 5228–5235, 2004.
- [48] N. Niu, J. Savolainen, T. Bhowmik, A. Mahmoud, and S. Reddivari, "A framework for examining topical locality in object-oriented software," in *Proceedings of the Annual Computer Software and Applications Conference (COMPSAC)*, 2012, pp. 219–224.
- [49] N. Niu and A. Mahmoud, "Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2012, pp. 81–90.

Automated Extraction and Visualization of Quality Concerns from Requirements Specifications

Mona Rahimi

School of Computing
DePaul University
Chicago, IL, 60604, USA
m.rahami@acm.org

Mehdi Mirakhori

School of Computing
DePaul University
Chicago, IL, 60604, USA
mehdi@cs.depaul.edu

Jane Cleland-Huang

School of Computing
DePaul University
Chicago, IL, 60604, USA
jhuang@cs.depaul.edu

Abstract—Software requirements specifications often focus on functionality and fail to adequately capture quality concerns such as security, performance, and usability. In many projects, quality-related requirements are either entirely lacking from the specification or intermingled with functional concerns. This makes it difficult for stakeholders to fully understand the quality concerns of the system and to evaluate their scope of impact. In this paper we present a data mining approach for automating the extraction and subsequent modeling of quality concerns from requirements, feature requests, and online forums. We extend our prior work in mining quality concerns from textual documents and apply a sequence of machine learning steps to detect quality-related requirements, generate goal graphs contextualized by project-level information, and ultimately to visualize the results. We illustrate and evaluate our approach against two industrial health-care related systems.

Index Terms—requirements, goal Model, quality concerns, visualization

I. INTRODUCTION

Software requirements come in many different shapes and sizes ranging from user stories to more traditional formats centered around “shall” statements. Their purpose is to clearly define the required functionality of the system and also to specify any quality constraints. However, in practice, requirements specifications tend to focus on the functional needs of stakeholders and often fail to describe their underlying quality concerns [2], [6], [8], [12]. Such concerns cover a broad range of qualities such as security, performance, availability, extensibility, and portability [8], [30] and play a critical role in the architectural design of the system.

When quality related requirements do exist, they are often incomplete and intermingled with functional requirements. The problem seems to arise from the underlying misconception that developers and customers have a shared, often unspoken, vision of the quality concerns of the system. However, this is often not the case and as a result, the delivered system may underperform against users expectations [2].

Having the ability to fully understand the quality concerns of a system and to identify missing or incomplete quality related requirements is especially important in domains such as the healthcare industry, governed by privacy concerns [4], [22], or in systems where accessibility, safety, or security concerns are regulated [29].

In this paper we present a novel approach for extracting and visualizing quality concerns from an existing set of requirements. Our approach, which is illustrated in Fig. 1, builds upon our prior work in using data mining techniques to detect non-functional requirements [9], and then introduces a novel approach for contextualizing the concerns according to specific domain topics. Instead of simply returning a flat list of quality concerns, we compose them into a meaningful hierarchy of topics. Our process includes three phases of modeling and detecting quality concerns, discovering impacted areas of functionality, and discovering and generating a meaningful hierarchy.

Visualizing concerns in such a hierarchy provides potential support for several important software engineering activities including: (1) understanding quality related concerns for the system in order to support compliance verification, (2) identifying underspecified concerns, (3) driving architectural design and assessment, (4) communicating with customers, and (5) supporting change impact analysis.

Throughout this paper we illustrate and evaluate our approach against two requirements specifications taken from the healthcare domain. The first, dataset is developed by the Certification Commission for Healthcare Information Technology (CCHIT) [5] and includes 235 requirements for managing electronic healthcare records. The second dataset represents 1,736 requirements extracted from documentation for World-Vista, the USA Veterans Health Care system [16]. We provide examples of requirements from each of these datasets in Table I along with the concerns they address.

The remainder of the paper is laid out as follows. Section II introduces the notion of a Softgoal Interdependency Graph (SIG) and discusses the specific challenges for automating its construction. Section III describes our existing approach for retrieving quality-related requirements from a specification. Section IV describes our solution for discovering topics, constructing a hierarchy of contextualized requirements, and generating the SIG. Sections V and VI then report on a user evaluation of the generated solution, and discuss several different usage scenarios. Finally, Section VII describes related work, Section VIII discusses threats to validity and Section IX summarizes our findings and proposes ideas for future work.

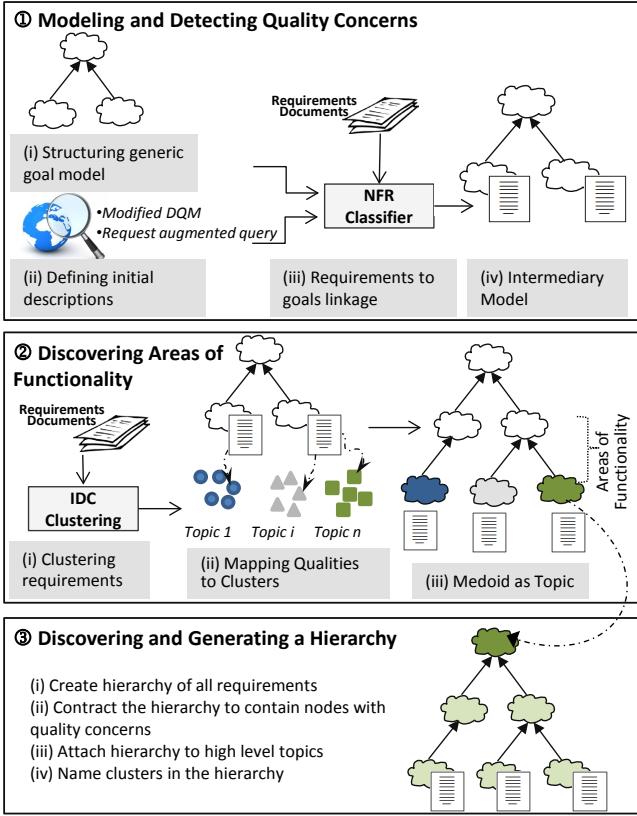


Fig. 1. Overview of the SIG construction and visualization process

II. GOAL MODELS

Our approach is designed to automatically generate a basic goal model from the requirements specification. We adopt the notation of Softgoal Interdependency Graphs (SIG), which are designed to model quality concerns and their various tradeoffs [8], [12]. While there are several approaches for modeling goals, including KAOS [36] and i* [37], we selected the SIG approach because it matched our objectives of visualizing a simple hierarchy of quality goals.

A. Softgoal Interdependency Graph

A SIG provides the graphical notation needed to depict the quality concerns of a system. Each concern is modeled as a softgoal, reflecting the fact that extensive interdependencies exist between various quality concerns, and furthermore, that trade-offs must be made in order to satisfice (i.e. sufficiently satisfy) each of the goals.

Fig. 2 provides an example of a security-related integrity goal for a customer-serving application. In this example, the *Integrity* softgoal is decomposed into *completeness* and *accuracy* subgoals. *Accuracy* is refined into *billing* and *patient information*, and *billing* is further refined into *accounts receivable* and *accurate invoices*. Finally, *nightly batch reconciliation* and *transaction controls* are proposed as design solutions (referred to as operationalizations) for achieving accurate invoices. The SIG also depicts contribution structures

TABLE I
SAMPLE REQUIREMENTS FROM WORLDVISTA AND CCHIT SHOWING
THEIR QUALITY TYPE

| Project | Sample Requirement | Type |
|---------|---|------|
| WV | Generates progress notes. Displays all information at the time of an ART event on the Progress Notes API and allows editing of the note prior to sign off. | F |
| WV | VistA Esig applications are required to authorize and authenticate their users. | A |
| WV | The system shall audit sensitive records, enabling a site to follow up on a record accessed by a remote location. | AU |
| CCHIT | The system shall provide the ability to capture and store discrete data regarding symptoms, signs and clinical history, from a clinical encounter and to associate that data with codes from standardized nomenclatures. | F |
| CCHIT | The system shall enforce the most restrictive set of rights/privileges or accesses needed by users groups (e.g. System Administration, Clerical, Nurse, Doctor, etc.), or processes acting on behalf of users, for the performance of specified tasks. | AC |
| CCHIT | The system upon detection of inactivity of an interactive session shall prevent further viewing and access to the system by that session by terminating the session, or by initiating a session lock that remains in effect until the user reestablishes access using appropriate identification and authentication procedures. The inactivity timeout shall be configurable. | AL |

F=Functional, A=Authentication, I=Integrity Controls
AUD=Audit, AC=Access, AL=Automated Logoff

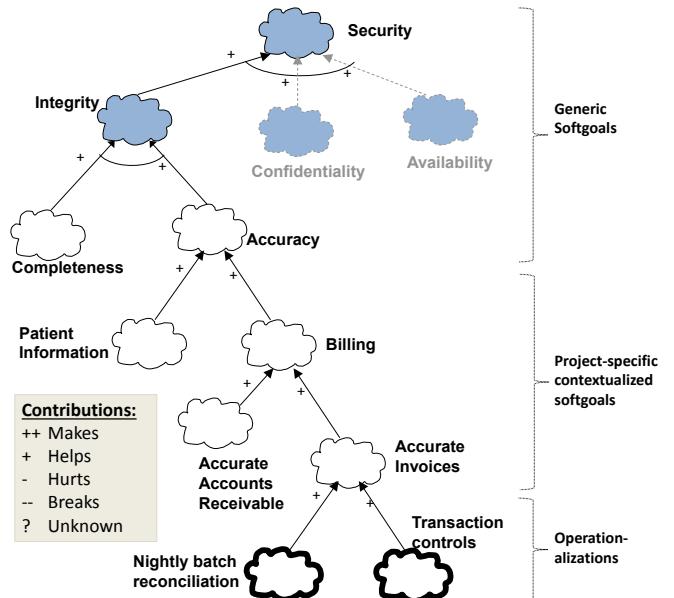


Fig. 2. An Example of a Softgoal Interdependency Graph (SIG) showing generic softgoals, contextualized softgoals, and candidate operationalizations

which capture the extent to which a child goal contributes to the satisficing of its parent. Each contribution is labeled as ++ makes, + helps, - hurts, and -breaks. A SIG would typically also include several candidate operationalizations and a variety of contribution structures. However, the reverse engineering

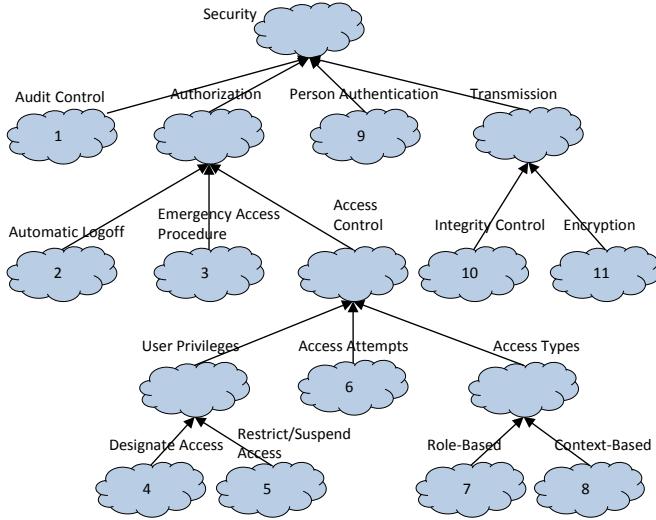


Fig. 3. Security concerns specified as a SIG scoping model define the SIG to be generated. Leaf nodes are numbered.

process presented in this paper seeks to reconstruct a current snapshot of the goals, quality concerns, and requirements, and therefore is not designed to reconstruct alternate design decisions or to model negative contributions.

B. Visualization Challenge

Generating a SIG automatically from a set of requirements is clearly a non-trivial multi-step task. As depicted in Fig. 1, the first phase involves the modeling of quality concerns and the classification and retrieval of related requirements from the specification. In phase 2, topics are identified and attached as subgoals to their relevant quality concerns. Finally, in phase 3, requirements associated with each topic are organized into a lower level hierarchy, thereby generating a candidate SIG. In the following sections we discuss each of these steps in greater detail.

C. Scope the Goal Graph

SIGs can be used to represent a wide variety of quality concerns, therefore it is necessary to clearly specify which quality concerns are to be included in the generated graph. This is accomplished through creating a *SIG Scoping Model* which defines the uppermost nodes of the SIG. For purposes of our case study we focus on security-related goals extracted from the US 1996 Health Insurance Portability and Accountability Act (HIPAA) technical safeguards. These goals, which are depicted in the SIG Scoping Model shown in Fig. 3, focus primarily on access controls including user authentication, activity audits, time-out controls, and the encryption and decryption of critical data.

III. IDENTIFY QUALITY-RELATED REQUIREMENTS

In the next step of the process, the requirements specification is searched for any requirement that is related to any of the quality concerns defined in the SIG Scoping Model.

This is accomplished through manually creating an initial search query for each leaf node. For example, as depicted in Table II, the leaf node *audit control* representing goal $G1$ is augmented to read *audit control security*. To further improve the quality of the search query, we enhance it using query augmentation [23]. This provides two primary benefits. First it automatically identifies a more extensive set of search terms and secondly it weights each of those terms according to their importance for the search. We utilize a tool that we had previously developed to augment trace queries [23]. This tool (1) utilizes the initial search query and a selection of standard search engines (Google, Yahoo, Bing etc) to retrieve a set of relevant documents, (ii) filters the results to remove documents which are difficult to parse (e.g, image-only files), (iii) splits each document into chunks of overlapping text, (iv) uses the basic vector space model cosine similarity approach to identify the chunk most relevant to the search query, (v) identifies and extracts domain specific terms and phrases which serve as candidate indicator terms, (vi) computes relevance metrics for each term (or phrase), and finally (vii) uses the metrics to identify the most relevant terms to serve as quality-type indicators.

TABLE II
AUGMENTED GOAL DESCRIPTIONS USED TO TRAIN THE REQUIREMENTS CLASSIFIER AND THE FINAL QUERY TERMS

| | Augmented Goal | Terms (weights not shown) |
|----|--|--|
| 1 | audit control security | secur, audit, system, secur audit, procedur, practic, govern, risk, center |
| 2 | automatic log off authorization | author, authent, payment, bank, comment, post, comput, login, click |
| 3 | emergency access procedure authorization | emerg, procedur, author, health, care, plan, provid, access procedur, secur |
| 4 | designate access user privilege | privileg, administr, account, permis, role, databas, owner, tabl, click |
| 5 | restrict suspend access user privilege | privileg, account, restrict, administr, databas, right, agent, logonid |
| 6 | access attempt access control | attempt, access control, access attempt, connect, network, polici |
| 7 | role based access type | role, access control, secur, rbac, permis, polici, type, resourc, server |
| 8 | context based access type | context, access control, secur, role, author, cbac, polici, cisco, decis |
| 9 | person authentication security | person, authent, secur, password, biometr, ident, method, person authent |
| 10 | integrity control transmission | secur, network, transmiss secur, integr control, transmiss integr, entiti, system, authent |
| 11 | encryption transmission | encrypt, data transmiss, secur, kei, transmiss encrypt, peer, connect, mac |

Table II shows the top-ranking indicator terms which were discovered using query augmentation for each goal. Terms are displayed in their root forms. Applying this technique to each leaf node produces a modified description of each associated goal, represented as a set of weighted terms.

The weighted indicator terms for each goal g are then used to evaluate the likelihood ($Pr_g(r)$) that a given requirement r is associated with the goal. Let I_g be the set of indicator terms for goal g identified during the query augmentation phase. The classification score that requirement r is associated with goal

g is then defined as follows:

$$Pr_g(r) = \frac{\sum_{t \in r \cap I_g} Pr_g(t)}{\sum_{t \in I_g} Pr_g(t)} \quad (1)$$

where the numerator is computed as the sum of the term weights of all type g indicator terms that are contained in r , and the denominator is the sum of the term weights for all type g indicator terms. The probabilistic classifier for a given type g will assign a higher score $Pr_g(r)$ to requirement r that contains several strong indicator terms for g .

This classification step attaches a list of relevant requirements to each quality-related leaf node in the SIG Scoping Model.

IV. CREATE HIERARCHY OF TOPICS

Each set of requirements is then organized into a meaningful hierarchy of topics. This kind of topical breakdown is common in practice. For example, the *accuracy* goal in Fig. 2 is decomposed into topical subgoals related to *patient information* and *billing* – implying that there are accuracy concerns in these two functional areas of the system.

We adopt two separate clustering techniques. The first identifies dominant topics of relevance to a specific quality concern. It uses the Incremental Diffusive Clustering (IDC) technique which has been shown to produce highly cohesive clusters of requirements [18], [24]. However, while IDC produces high quality clusters, it does not place every requirement into a cluster. Therefore we implement a second lower-level clustering which creates a sub-hierarchy of topics underneath each of the IDC-identified topics. Each clustering technique requires its own naming process, and additional steps are needed to generate the complete SIG. Given the multi-step process, we follow a workflow of data-mining techniques.

A. Step 1: Identify primary topics

Dominant topics are identified through the use of IDC. This is an incremental process, in which a simple clustering technique, such as SPK-Means is used to cluster all of the requirements into k clusters. The number k can be determined either through trial-and-error (i.e. by observing the output to determine if the clusters are sufficiently cohesive and at an appropriate level of granularity), or as in our case, through applying an information-entropy based metric such as Can's Metric [24]. Following each round of SPK-Means clustering, the quality of each cluster is evaluated according to its size and cohesion – with the goal being to identify highly-cohesive, non-trivially sized clusters. The ‘best’ cluster is then selected, and its key terms are identified and removed from all requirements in the dataset. The IDC clustering process is then repeated against the reduced requirements. The process is repeated and one new topic (i.e. one cluster) is output following each iteration. A more complete description of IDC is provided in our prior work [24].

As the IDC clusters need to be displayed to the user, it is important to provide meaningful names for each topic. The common practice is either to substitute the most prominent

terms (i.e. a bag of words) for the name, or to select a recurring phrase from the clustered requirements. Our approach builds upon the second option and involves first identifying the cluster’s medoid, defined as the term or phrase which is most representative of all requirements in the cluster, and then removing quality-related terms from the medoid of each cluster. These are the terms from the initial search query such as the terms *encrypt*, *data transmiss*, *secur*, *kei*, taken from the *encryption* goal etc. Stop words, i.e. very common words such as *this* and *or* are also removed. The highest scoring remaining term is then used to seed the topic name. Each seed term is then traced back into the requirements of the cluster and its prior- and post terms are attached to the *medoid* terms. In the case of having multiple compound terms, the term with the highest frequency is selected. The final compound term selected for each of the encryption topics is shown in the fourth column of Table III. For example, topic # 25 is named *enhance security*.

B. Step 2: Attach Topics to SoftGoals

Topics are attached to a softgoal if the IDC generated cluster contains two or more requirements related to that softgoal. For example, in Table III, the topic related to *client-server* has four requirements classified as *encryption*-related, and is therefore attached to the *encryption* goal. In contrast, the *printed form* topic contains only a single requirement and therefore is not attached. We refer to this requirement as a singleton. Unattached requirements include all singletons, plus any quality-related requirements not placed into a cluster by IDC. We explored two different approaches for placement of these requirements into the hierarchy, and found that the CHITT dataset performed best with the first approach, and World Vista with the second. While our goal is to identify an approach which works consistently across all datasets, for now we report both techniques.

In the first approach, all unattached requirements were initially assigned to each of the two identified topics and the assignment decision was deferred for later in the process. In the second approach, Latent Dirichlet Allocation (LDA) was used to place each requirement into the topic with the most contextually similar requirements. LDA is a generative probabilistic model that discovers short descriptions (topics) for the members of a large collection [3]. It estimates the similarity between each requirement and the discovered topics and was therefore used to identify the best sub-goal placement for each of the unattached requirements. We implemented it using the Stanford Topic Modeling Toolbox with default parameters [28].

C. Step 3: Build a Sub-Hierarchy

At this stage in the workflow, we have high level topics assigned to each quality concern.

To create the lower part of the SIG, we used Agglomerate Hierarchical Clustering (AHC) because it explicitly constructs a hierarchy of topics. It works on the term vector representation of the requirements previously created and utilized

TABLE III

CLUSTERS IDENTIFIED FOR THE *encryption* GOAL IN WORLD VISTA. THE FIRST COLUMN DEPICTS THE MEDOID OF EACH CLUSTER. IN COLUMN 2 THESE ARE FILTERED TO REMOVE STOPWORDS AND SEARCH TERMS, AND IN COLUMN 3 THE HIGHEST SCORING REMAINING TERM IS EXPANDED (WHERE POSSIBLE) INTO A MORE DESCRIPTIVE PHRASE

| ID | Topics extracted from IDC | Filtered | Compound Term | Count |
|----|---|----------|---------------------|-------|
| 16 | client, server, broker, rpc, vist, connect' | client | client/server | 4 |
| 27 | entri, clerk, data' | entri | clinician data | 2 |
| 13 | 'form, encount' | form | printed form | 1 |
| 21 | 'site' | site | site-specific | 1 |
| 22 | 'updat, demograph, patient' | updat | updated demographic | 1 |
| 25 | 'secur, enhanc' | enhanc | enhance security | 1 |
| 28 | 'mail, group' | mail | MailMan | 1 |
| 35 | 'ad, ami, hoc' | ad | AD Hoc | 1 |

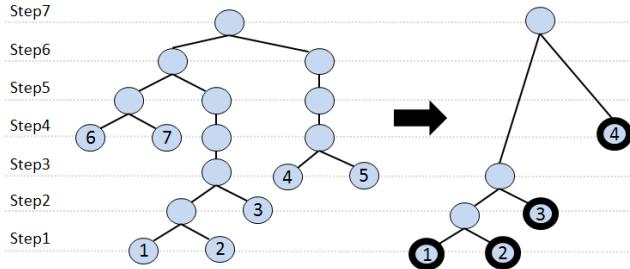


Fig. 4. Result of Agglomerate Hierarchical Clustering. Topic related requirements are found in clusters 1, 2, 3 and 4

for classification purposes. AHC is a bottom-up hierarchical approach that starts by placing each requirement into an individual cluster, and then in each subsequent iteration, merges the two clusters which are most similar using a standard similarity metric and a linkage method. We adopted Squared Euclidean Distance and Ward's method to link the most similar clusters. In Ward's method the two most similar clusters are those for which the merger result has the minimum increase in the error sum of squares [15].

Hierarchical clustering techniques produce dendograms in which clusters are connected at different levels of the hierarchy. Our goal is to create a partition of disjoint clusters to visualize a meaningful hierarchy of contexts. To determine the cutting point we selected the point at which the gap, measured by similarity scores, between two successive combinations of subtrees is the largest. In effect, splitting clusters any further would decrease their quality [7].

While it might intuitively make sense to apply AHC to only those requirements associated with each higher-level topic this is impractical when only a few requirements are involved. We therefore applied AHC to all of the requirements in the specification creating a complete hierarchy of topics. The nodes in the hierarchy which contain relevant requirements are then extracted from the general hierarchy and rejoined to form a proper subtree. This is illustrated in Fig. 4. The left hand side shows the complete hierarchy of WorldVista requirements, while the right hand side of the same figure

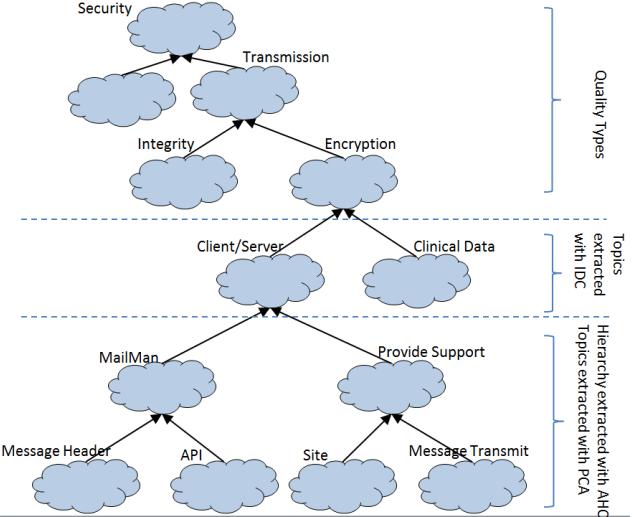


Fig. 5. Steps of Contextualizing Quality Concerns for Encryption subgoal of Transmission Security

represents the sub-hierarchy for a security concern subgoal *Transmission Integrity*. Its requirements are located in clusters 1, 2, 3 and 4. We prune any nodes which neither contain a requirement relevant to the topic, nor are located on a direct path from such a node to the root. Finally, any linear sequence of nodes without siblings are merged into a single node. In this manner the subset of relevant nodes are extracted from the complete hierarchy.

Once again, we need to label each of the newly identified nodes in a meaningful way. Unfortunately, the medoid approach does not work well on the smaller number of requirements assigned to each subnode, and therefore we adopted a different approach based on the use of Principal Component Analysis (PCA). PCA is a feature extraction method which transforms a set of features to a smaller set of attributes called Principal Components (PC). We applied PCA against the vector of terms representing each requirement in an individual cluster with the goal of extracting the underlying structure of the data [1]. While PCs are discovered incrementally, the first PC captures as much of the variability in the data as possible. Each succeeding component captures as much of the remaining variability as possible. Mathematically, this transformation is defined by a set of d-dimensional vectors of weights or *loadings* $w_{(i)} = (w_1, \dots, w_p)_{(i)}$ that map each original feature $F_{(j)}$ of F to a new vector of principal component scores $PC_{(j)} = (PC_1, \dots, PC_p)_{(j)}$. Below is the general formula for the first PC:

$$PC_1 = w_{11}(F_1) + w_{12}(F_2) + \dots + w_{1p}(F_p) \quad (2)$$

The loadings represent how much the original feature has contributed in forming the related PC. The larger the loading, the greater the contribution of the feature in forming the associated PC. For example, w_{11} in equation 2 represents the contribution of the original term F_1 in forming PC_1 .

We applied PCA to the set of requirements in the cluster, identified the first PC, and then selected the terms with the

TABLE IV
RECALL AND PRECISION ACHIEVED FOR REQUIREMENTS CATEGORIZATION

| Goal | Name | Count | DQM | | | Augmented | | | Basic | | | |
|------|-------------------------|-------|-----------|------|------|-----------|------|------|-------|------|------|------|
| G1 | audit control | 26 | Precision | 0.75 | 0.58 | 0.65 | 0.72 | 0.50 | 0.59 | 0.25 | 1.00 | 0.40 |
| G2 | automatic log off | 2 | Precision | 0.50 | 0.50 | 0.50 | 0.77 | 1.00 | 0.87 | 0.72 | 0.50 | 0.59 |
| G3 | emergency access | 6 | Precision | 1.00 | 0.83 | 0.91 | 1.00 | 0.50 | 0.67 | 0.00 | 0.00 | 0.00 |
| G4 | designate access | 1 | Precision | 1.00 | 1.00 | 1.00 | 0.50 | 1.00 | 0.67 | 1.00 | 1.00 | 1.00 |
| G5 | suspend access | 9 | Precision | 0.71 | 0.56 | 0.63 | 0.20 | 0.25 | 0.22 | 0.50 | 0.50 | 0.50 |
| G6 | access attempt | 1 | Precision | 1.00 | 1.00 | 1.00 | 0.63 | 1.00 | 0.77 | 0.83 | 1.00 | 0.91 |
| G7 | role based access | 7 | Precision | 1.00 | 1.00 | 1.00 | 0.71 | 0.57 | 0.63 | 0.50 | 0.14 | 0.22 |
| G8 | context based access | 2 | Precision | 0.50 | 1.00 | 0.67 | 0.11 | 0.50 | 0.18 | 0.00 | 0.00 | 0.00 |
| G9 | integrity control | 7 | Precision | 0.57 | 0.71 | 0.63 | 0.50 | 0.28 | 0.36 | 0.50 | 0.28 | 0.36 |
| G10 | encryption transmission | 4 | Precision | 0.25 | 0.25 | 0.25 | 0.33 | 0.50 | 0.40 | 0.25 | 0.50 | 0.33 |

largest loading. For instance, the PC vector below represents the first PC for the context *Message Header* located in Fig. 5.

$$PC_1 = 0.598header + 0.276includes + 0.187generate\dots$$

As shown above, the term associated with the largest loading is *header*. In order to retain the meaning of compound terms, we search for the selected term in the requirements description and return the phrase in which the term most frequently appears.

Furthermore, in our first approach for placement of unassigned requirements, we place a requirement into a topic if the terms in that requirement contribute to the first PC.

D. Step 4: Put it all together

Finally, the SIG is constructed by attaching topics to the lowest level of soft goals, and then by attaching the hierarchy of relevant requirements nodes under each of the topics. The SIG structure is generated as an XML file which can then be read into a SIG graphing tool. We have developed a prototype tool for this purpose.

V. EVALUATION

For evaluation purposes we conducted two different experiments. The first comparatively evaluated three classification techniques for retrieving requirements related to each of the specified soft goals, while the second evaluated the quality of the generated SIG itself.

A. Connecting Softgoals to Requirements

For purposes of the first experiment we constructed an ‘answer set’ for the CCHIT data. This answer set consisted of 235 requirements categorized according to quality concerns specified in Fig. 3.

1) *Experimental Design*: Three different query types were used to classify requirements. The *basic* approach used the initial terms describing each goal. The *augmented* approach used the web-based search technique to augment the query. We also compared a third more interactive approach in which the user built the query iteratively as he/she reviewed the results. This approach is aligned with a technique referred to as Direct Query Manipulation (DQM) which has been used effectively for trace retrieval purposes [33]. While this third approach requires human interaction, it is highly appropriate in a context

for which it makes sense for the user to interact with, and refine the search results.

2) *Results*: Results from this experiment are reported using standard metrics of *recall* (i.e. the fraction of correct requirements which were retrieved), *precision* (i.e. the fraction of retrieved requirements which are correct), and F-Measure which computes the harmonic mean of recall and precision. These results show that DQM performed the best with a mean F-Measure of 0.72, compared to 0.54 for the augmented search query, and 0.43 for the basic approach. Furthermore, DQM returned the best results in 8 out of 10 cases (tying with basic in one case). Comparing the two fully automated approaches, Augmented outperformed Basic in six cases, and tied in one case. These results suggest that the initial search could be performed using the augmented search query provided by the user; however, if the user is not satisfied with the results she could use DQM to manually modify the augmented query.

B. Evaluating the Generated SIGs

The second experiment evaluated the quality of the SIGs generated for the CCHIT and WorldVista requirements. We first performed a preliminary analysis to determine the feasibility of manually creating SIGs that could serve as answer sets for evaluation purposes. However, it became obvious that there were many different viable ways to decompose the quality concerns and topics, and that no single *ground truth* model existed. This observation has also been made for constructing and evaluating graphs of feature models [13]. Therefore, the feasibility of evaluating our approach against a ground truth model was rejected. As a result, we conducted a qualitative evaluation.

C. Qualitative Evaluation

To evaluate the quality of the generated SIG we asked eight software engineers from the Professional MS Software Engineering program at DePaul, to assess the quality of the models. A SIG was generated for each of the datasets using the SIG Scoping Model depicted in Fig. 3. Given the size of each dataset, the number of nodes in the generated SIG, and the number of quality-related requirements attached to each node, we used only a sample of requirements and generated nodes for the evaluation.

TABLE V
SAMPLE OF THE QUALITATIVE EVALUATION

| Goal Hierarchy | Requirement Specification | Evaluation |
|---|--|--|
| Audit Control>Patients Record>Audit Record> | The system shall require documentation of the audit support functionality in the vendor provided user guides and other support documentation, including how to identify and retrospectively reconstruct all data elements in the audit log including date, time. | All match |
| Authorization>Access Control>Restrict, Suspend Access>Standard base>Designated Patient's Chart> | The system, prior to a user login, shall display a (configurable) notice warning (e.g. The system should only be accessed by authorized users). | Only related to the sub-goal |
| Authorization>Automatic Logoff>HIR>Medical Record> | The system shall support medical record entries that are dated, the author identified and when necessary according to law or regulation or hospital policy, authenticated, either by written signature, electronic signature or computer key. | Only related to the project-specific context |

TABLE VI
RESULTS OF QUALITATIVE EVALUATION

| | CCHIT | Unrelated | WorldVista | Unrelated |
|--|-------|-----------|------------|-----------|
| Not Related at all | 0.11 | 0.68 | 0.08 | 0.37 |
| Only related to the sub-goal | 0.27 | 0.12 | 0.10 | 0.12 |
| Only related to the project-specific context | 0.16 | 0.04 | 0.05 | 0.06 |
| All match | 0.46 | 0.16 | 0.76 | 0.44 |

However, in order to provide coverage of all goal types, requirements assigned to nodes were selected using *random stratified sampling* in which samples were randomly taken from each node in a number proportional to the size of the group and independent from each other. Furthermore, in order to test for participant bias, to minimize the impact of the lack of domain knowledge, and to proactively encourage users to intelligently evaluate the results, we included a set of randomly selected unrelated requirements into the SIG. The participants were informed that some percentage (unspecified) of the requirements had been assigned randomly to nodes.

Prior to the study, one of the researchers gave a one hour presentation about SIGs and quality related requirements, in order to familiarize participants with the approach. Participants were then randomly assigned to either the CCHIT or WorldVista group, such that four people evaluated each goal model.

Each participant was given a copy of one of the goal models and its associated sample requirements and was asked to rate the correctness of the requirements placement in the model, and the relevant hierarchy based on the following criteria:

- 1) *Not Related at all*, The Requirement is misclassified. It neither belongs to the security sub-goal nor is relevant to the context. (Score = 1).
- 2) *Related to the sub-goal* The requirement is correctly classified under a security subgoal but does not relate to the project-specific context. (Score = 2).
- 3) *Related to the project-specific context*, The requirement is correctly assigned to a topic in the project-specific context, but does not relate to the security sub-goal. (Score = 3).
- 4) *Complete match*, The requirement is correctly placed under an appropriate sub-goal and is also correctly related to the project-specific context. (Score = 4).

Table V presents three detailed examples of extracted sub-goals. The first example shows a requirement which is correctly classified as *Audit Control* and also correctly contextualized around the domain concept of *Patient Record*. The second example shows a requirement which is correctly contextualized within its system function but is misclassified under a wrong security sub-goal. Lastly, the third example shows a correct classification under the *Restrict, Suspend Sub-Goal* but which is not presented within a meaningful project-level context.

D. Analysis of Results

An analysis of the results showed that our approach performed better for the World Vista dataset than for CCHIT. In the case of WorldVista, 76% of the generated nodes were graded as correctly placed, but 44% of the unrelated nodes were also graded this way. In the case of CHITT the ratio was 46% vs. 16%. Furthermore, in CCHIT an additional 43% of requirements were placed only partially correctly, with 16% for WorldVista.

One explanation for these less than stellar results is the fact that our SIG Scoping Model includes closely related quality concerns, making it difficult to decide which node a given requirement is most closely related to. For example, a requirement describing role-based access control to restrict access to a specific functional area could equally well fit under nodes 4 or 7 in the SIG Scoping Model depicted in Fig. 3. This suggests that future attempts to generate SIGS should focus on coarser grained categories of concerns.

In conclusion, while our approach performs better than the random baseline in both cases, the generated SIG is far from perfect and not yet at a sufficient quality for practical use. Nevertheless, we present our results as an initial baseline for further work in the area.

VI. APPLICATIONS

Earlier in the paper we made several claims concerning the benefits that SIG visualization, if achieved successfully, could provide for various Software Engineering activities. In this section we briefly support these claims using illustrations from the SIG generated for the WorldVista project.

A. Understanding quality related concerns

Understanding the specified non-functional requirements of the system is critical for downstream design purposes and also for compliance verification. We illustrate this in Fig. 6 with a visualization that depicts the extent to which each soft goal in the upper part of the SIG is covered by requirements. In this visualization we annotate each goal node with the count of associated requirements, and use a shading scheme which depicts nodes with more requirements in darker shades and those with fewer requirements in lighter shades. Nodes with no identified coverage are shown with dashed lines. Tool support could allow an analyst to explore the lower levels of the SIG to see the expanded hierarchy of these requirements. Such views enable an analyst to answer questions such as “to what extent is quality concern Q specified in the requirements?” or “Which parts of the system are affected by quality Q?” In many projects, for which requirements are listed only as flat files, this kind of perspective is only feasible if the goal view is generated automatically.

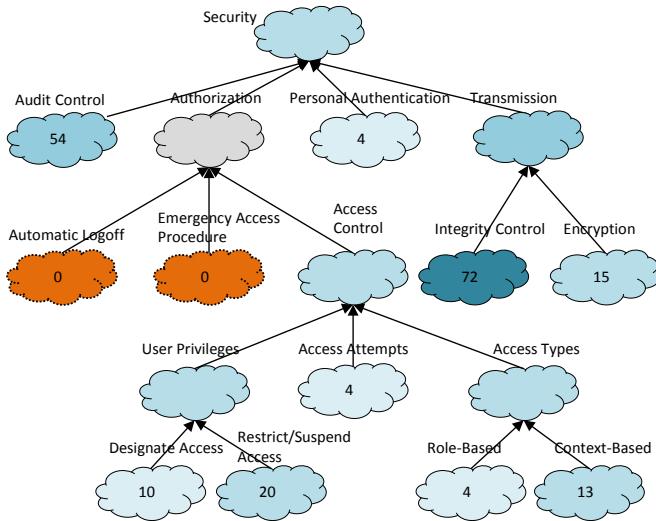


Fig. 6. A Visualization of the Overall Security Coverage for World Vista

B. Identifying Underspecified Concerns

In Figure 6 two of the concerns, namely *automatic logoff* and *emergency access procedures* are clearly underspecified with no associated requirements. An analyst could explore this more closely. First, he might modify the initial search terms for the goal, either manually, or through requesting a query augmentation, and then reexecute the search. If no additional requirements are retrieved, then the analyst would need to investigate whether the goal is relevant or not, and

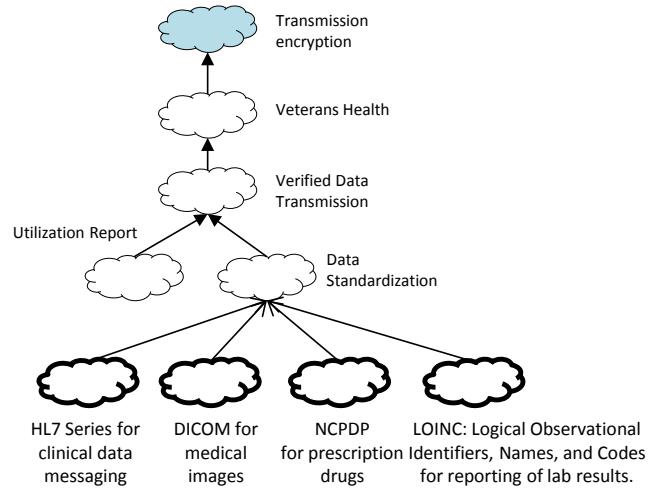


Fig. 7. Attaching candidate design decisions to the goal graph. These candidate operationalizations are shown with bold borders at the leaf nodes of the SIG.

if not, whether missing requirements should be specified. In the case of World Vista, both of the underspecified goals are required by HIPAA and therefore likely should be specified as requirements.

C. Driving design and assessment

Goal graphs explicitly model softgoals and therefore provide the opportunity for proposing and evaluating various design decisions. For illustrative purposes consider the goal to achieve *verified data transmission*. This goal contributes to achieving veterans health and privacy through transmission encryption. The generated goal graph provides a visual environment in which to reason about design solutions. Four candidate solutions are shown for consideration in the SIG. These propose various standards for transmitting general health information, medical images, prescription drug information, and laboratory results. These candidate operationalizations can be manually added to the automatically generated goal graph as depicted in Figure 7 and used to reason about the contribution of various options.

D. Supporting Change Impact Analysis

It is well documented that long-term changes to code often result in inadvertent degradation of quality concerns [20]. Reverse engineering the SIG from requirements and mapping quality concerns to various functional topics in the system introduces the potential for providing information about quality concerns during change impact analysis. For example, we can extract information from the generated SIG such that *blood bank* has role-based access requirements or that *clinician entries* must be encrypted before transmission. When new requirements are introduced to a part of the system, the SIG provides the means of identifying relevant quality concerns.

E. Communicating with Customers

Much software development effort occurs in existing systems. Customers request new features and/or modifications

to existing ones. By reverse engineering a SIG we are able to discuss new features in light of existing quality concerns, thereby increasing the likelihood of delivering a satisfactory product.

VII. RELATED WORK

There have been several attempts to develop visualization techniques for quality concerns. These techniques range from simple tabular and relational formats to hierarchical, quantitative, and metaphorical approaches [10]. The majority of these approaches focus on visual notations for capturing concerns, and for presenting relationships between different aspects. However, there is limited work on automated extraction and visualization of software requirements, especially related to non-functional qualities.

Goal-oriented approaches capture and visualize non-functional concerns [19]. Such techniques include the soft-goal interdependency model [8], the i^* framework [37], and KAOS [36] as well as more general approaches based upon visualizing quality concerns across UML diagrams [35], [11].

However, such approaches tend to focus on capturing requirements knowledge without providing automated support. Some exceptions are a visual modeling tool developed by Farid and Mitropoulos [21], which provided a semi-automated approach for capturing, visualizing, and reasoning about quality concerns in agile projects. They used a modified form of the SIG to assist agile developers during the requirements gathering and analysis phase.

ConcernMapper [31] and AoGRL [26] tools provide visualization support for aspect oriented software development. These tools attempt to depict the way in which cross-cutting concerns, such as security, logging, etc., are scattered across functional parts of a software system. However such approaches are not fully automated and the visualizations fail to provide detailed information about interactions between concerns and functional features. EAMiner, [32] utilizes natural language processing techniques, to mine various types of cross cutting concerns from requirements documents. The output is a flat modeling of the early aspects found in specification documents, and is therefore akin to the initial part of our process in which we retrieve a list of requirements associated with a specific goal.

Several researchers have utilized information retrieval techniques to extract quality concerns from software documents [34], [25], [9]; however, such approaches only focus on detecting and retrieving quality concerns and, like EA-Miner, return the requirements in the form of a flat list. In contrast our approach attempts to create a hierarchy of goals driven by the functionality of the system in which the goals are applied.

Finally, researchers [17], [27] have also utilized clustering algorithms to discover key functional features in software documents and then to visualize the requirements and their dependencies. These works have mainly focused on functional requirements and have not explored the visualization of quality concerns. Similarly Delfosse et al., [14] automated the visualization of a feature model for a software domain by first

discovering the domain features through clustering hundreds of partial product descriptions mined from SoftPedia and then using association rule mining and inferencing to generate the goal tree. The unique characteristic of their approach is the fact that the feature model was generated from thousands of different systems, enabling the use of techniques which are not applicable in a single project. Furthermore, their approach focused on functionality and generated a feature model used traditionally for product-line development, whereas the approach reported in this paper focuses on quality concerns.

VIII. THREATS TO VALIDITY

Threats to validity are classified as internal, construct and external validity.

Internal validity reflects the extent to which the bias or error has been reduced in the method so a causal relationship can be concluded from the study. In our work, in the evaluation process of the goal hierarchy, it was difficult to totally separate out the task of evaluating the replacement of requirements, with the quality of the associated topics generated for them. For instance, if we asked a user to evaluate whether the hierarchy of context $C1, C2, \dots$ was meaningful for requirement R , then we might bias the participant with both the essence of the topic for the related requirement(i.e., what the requirement truly represents) as well as the selected terms for the topic. In order to mitigate the bias, we embedded unrelated requirements into several of the nodes.

Construct validity discusses the quality of choices for the dependent and independent variables and evaluates the extent to which the construct that was intended to be measured was actually measured. In qualitative evaluation, although the participants have been selected from experts in software engineering they were familiar with, yet not experts in, the health care domain. On the other hand, the provided requirements were self-explanatory even to a non-expert. We minimized this error by providing a one hour introduction to SIGs showing some of the concepts from the domain. In future work we will repeat the evaluation using experts from the domain.

External validity is the generalizability of the approach. Although, we applied our method to two different large health care related systems exhibiting different characteristics of size and quality concerns, we will still need to extend our evaluation into other domains, and indeed into more projects within the current domain before we can claim generalizability.

IX. CONCLUSION

In this paper we have proposed an approach for automatically extracting a goal model from a requirements specification. The specific quality concerns of interest are defined manually by the analyst. We then implemented a sequence of machine learning and data mining methods to automatically detect quality concerns and to organize them into a meaningful hierarchy of topics.

Our evaluation demonstrated the viability of identifying and attaching requirements to quality concerns. However, we achieved mixed results in generating a meaningful hierarchy

of well-named goals and sub-goals. Further work is clearly needed to fine-tune our techniques so that each node of the graph is appropriately named, and that all functional areas of the system impacted by a specific quality concern are faithfully depicted in a well-structured hierarchy. We therefore plan to explore alternate goal decompositions and naming schemes in order to further increase the understandability and completeness of the generated SIG. Finally, we need to explore SIG generation across domains other than the healthcare industry, and explore a different and broader set of topics and quality concerns. We consider the solutions presented in this paper to be a baseline for future approaches.

Finally, we plan to extend our initial prototyping tool in order to create a more interactive environment in which the user can evaluate quality concerns for a project.

ACKNOWLEDGMENTS

This work was partially funded under National Science Foundation grant CCF-1218303.

REFERENCES

- [1] H. Abdi and L. J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.
- [2] D. Ameller, C. P. Ayala, J. Cabot, and X. Franch. Non-functional requirements in architectural decision making. *IEEE Software*, 30(2):61–67, 2013.
- [3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.
- [4] T. D. Breaux and A. Rao. Formal analysis of privacy requirements specifications for multi-tier applications. In *RE*, pages 14–20, 2013.
- [5] Certification Commission for Healthcare Information Technology. Guidelines for electronic healthcare records. <http://cchit.org>.
- [6] L. Chen, M. A. Babar, and B. Nuseibeh. Characterizing architecturally significant requirements. *IEEE Software*, 30(2):38–45, 2013.
- [7] H. S. Christopher D. Manning, Prabhakar Raghavan. Introduction to information retrieval. 1, 2008.
- [8] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Press, 2000.
- [9] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. The detection and classification of non-functional requirements with application to early aspects. In *RE*, pages 36–45, 2006.
- [10] J. Cooper, S.-W. Lee, R. Gandhi, and O. Gotel. Requirements engineering visualization: A survey on the state-of-the-art. In *Requirements Engineering Visualization (REV), 2009 Fourth International Workshop on*, pages 46–55, Sept 2009.
- [11] L. Cysneiros and J. Sampaio do Prado Leite. Nonfunctional requirements: from elicitation to conceptual models. *Software Engineering, IEEE Transactions on*, 30(5):328–350, May 2004.
- [12] L. M. Cysneiros and J. C. S. do Prado Leite. Nonfunctional requirements: From elicitation to conceptual models. *IEEE Trans. Software Eng.*, 30(5):328–350, 2004.
- [13] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans. Feature model extraction from large collections of informal product descriptions. In *ESEC/SIGSOFT FSE*, pages 290–300, 2013.
- [14] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans. Feature model extraction from large collections of informal product descriptions. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2013*, pages 290–300, New York, NY, USA, 2013. ACM.
- [15] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.
- [16] Department of Veterans Affairs Office of Enterprise Development. vista-healthvet monograph.
- [17] C. Duan and J. Cleland-Huang. Clustering support for automated tracing. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering, ASE '07*, pages 244–253, New York, NY, USA, 2007. ACM.
- [18] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. On-demand feature recommendations derived from mining public product descriptions. In *ICSE*, pages 181–190, 2011.
- [19] N. Ernst, Y. Yu, and J. Mylopoulos. Visualizing non-functional requirements. In *Requirements Engineering Visualization, 2006. REV '06. First International Workshop on*, pages 2–2, Sept 2006.
- [20] D. Falessi, L. C. Briand, G. Cantone, R. Capilla, and P. Kruchten. The value of design rationale information. *ACM Trans. Softw. Eng. Methodol.*, 22(3):21:1–21:32, July 2013.
- [21] W. Farid and F. Mitropoulos. Normatic: A visual tool for modeling non-functional requirements in agile processes. In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–8, March 2012.
- [22] A. Finnegan, F. McCaffery, and G. Coleman. Development of a process assessment model for assessing security of it networks incorporating medical devices against iso/iec 15026-4. In *HEALTHINF*, pages 250–255, 2013.
- [23] M. Gibiec, A. Czauderna, and J. Cleland-Huang. Towards mining replacement queries for hard-to-retrieve traces. In *ASE*, pages 245–254, 2010.
- [24] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher. Supporting domain analysis through mining and recommending features from online product listings. *IEEE Trans. Software Eng.*, 39(12):1736–1752, 2013.
- [25] I. Hussain, L. Kosseim, and O. Ormandjieva. Using linguistic knowledge to classify non-functional requirements in srs documents. In *Proceedings of the 13th International Conference on Natural Language and Information Systems: Applications of Natural Language to Information Systems, NLDB '08*, pages 287–298, Berlin, Heidelberg, 2008. Springer-Verlag.
- [26] G. Mussbacher, D. Amyot, J. Araujo, A. Moreira, and M. Weiss. Visualizing aspect-oriented goal models with aogrl. In *Requirements Engineering Visualization, 2007. REV 2007. Second International Workshop on*, pages 1–1, Oct 2007.
- [27] N. Niu and S. Easterbrook. On-demand cluster analysis for product line functional requirements. In *Proceedings of the 2008 12th International Software Product Line Conference, SPLC '08*, pages 87–96, Washington, DC, USA, 2008. IEEE Computer Society.
- [28] D. Ramage. The stanford topic modeling toolbox, Sept. 2009. <http://nlp.stanford.edu>.
- [29] P. Rempel, P. Mäder, T. Kuschke, and J. Cleland-Huang. Mind the gap: Assessing the conformance of software traceability to relevant guidelines. In *36th International Conference on Software Engineering (ICSE)*, 2014.
- [30] S. Robertson and J. Robertson. *Mastering the Requirements Process*. Addison-Wesley, 2013.
- [31] M. Robillard and G. Murphy. Concern graphs: finding and describing concerns using structural program dependencies. In *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, pages 406–416, May 2002.
- [32] A. Sampaio and A. Rashid. Mining early aspects from requirements with ea-miner. In *Companion of the 30th International Conference on Software Engineering, ICSE Companion '08*, pages 911–912, New York, NY, USA, 2008. ACM.
- [33] Y. Shin and J. Cleland-Huang. A comparative evaluation of two user feedback techniques for requirements trace retrieval. In *SAC*, pages 1069–1074, 2012.
- [34] J. Slankas and L. Williams. Automated extraction of non-functional requirements in available documentation. In *Natural Language Analysis in Software Engineering (NaturaLiSE), 2013 1st International Workshop on*, pages 9–16, May 2013.
- [35] S. Supakkul and L. Chung. Visualizing non-functional requirements patterns. In *Requirements Engineering Visualization (REV), 2010 Fifth International Workshop on*, pages 25–34, Sept 2010.
- [36] A. van Lamsweerde. Goal-oriented requirements engineering: A roundtrip from research to practice. In *RE*, pages 4–7, 2004.
- [37] E. S. K. Yu. Social modeling and i*. In *Conceptual Modeling: Foundations and Applications*, pages 99–121, 2009.

Language Extended Lexicon Points: Estimating the Size of an Application using Its Language

Leandro Antonelli

Lifia, Fac. de Informática
UNLP, Bs As, Argentina
lanto@lifia.info.unlp.edu.ar

Gustavo Rossi

Lifia, Fac. de Informática
UNLP, Bs As, Argentina
gustavo@lifia.info.unlp.edu.ar

Julio Cesar Sampaio
do Prado Leite

Dep. Informática
PUC-Rio, Gávea, RJ, Brasil
julio@inf.puc-rio.br

Alejandro Oliveros

INTEC
UADE, Bs As, Argentina
oliveros@gmail.com

Abstract—Estimating the size of a software system is a critical task due to the implications the estimation has in the management of the development project. There are some widely accepted estimation techniques: Function Points, Use Case Points and Cosmic Points, but these techniques can only be applied after the availability of a requirements specification. In this paper, we propose an approach to estimate the size of an application previous to its requirements specification by using the application language itself, captured by the Language Extended Lexicon (LEL). Our approach is based on Use Case Points and on a technique which derives Use Cases from the LEL. The proposed approach provides a measure of the application's size earlier than the usual techniques, thus reducing the effort needed to apply them. An initial experiment was conducted to evaluate the proposal.

Index Terms—*Requirements specifications, Domain Analysis, Language Extended Lexicon, Use Case Points, Software Sizing.*

I. INTRODUCTION

Estimating the size of a software system is a critical task since this estimation is used to plan the software system construction. Function Points [2], Cosmic Points [12] and Use Case Points [18] are three widely accepted estimation techniques. Function Points allow us to estimate the size of an application from the decomposition of the application into basic functions and files. Cosmic Points analyze functions that input, store, retrieve and output data. Use Case Points (UCP) rely on Use Cases (UC) to estimate the software size. Though these techniques have been widely used in practice, they are being adjusted continuously [17] [10] [19] [24] [12].

In order to apply any of these techniques, we need to elicit requirements to gain a deep understanding of the application, so that we can identify functions or Use Cases, then write them and afterwards perform the counts. These techniques can only be applied after requirements elicitation, analysis, and specification have been carried out. But, generally, projects are agreed upon previously to the existence of a full requirements specification.

Thus, we need an approach which can estimate the size of the software at a very early stage and obtain an objective measure, i.e., we need an approach which allows the project manager to obtain a preliminary estimation with low effort in order to use this information to sign a contract.

In this paper, we propose an approach to estimate the size of an application prior to requirements specification. We use

the Language Extended Lexicon (LEL) [21] to capture the application language and we analyze it to estimate the application's size. In previous works it has been shown that it is possible to identify ontologies [9], crosscutting concerns [5] and requirements [4] from the LEL. Use Cases can be derived from the LEL [4] and Use Case Points can be applied to these Use Cases [18]. Thus, inspired by Use Case Points and considering Use Case derivation from the LEL, we set out to develop the proposed strategy. Our approach analyzes the LEL in much the same way as Use Case Points analyze Use Cases, considering the correlation between the LEL and Use Cases in terms of the derivation strategy proposed in [4]. It is important to mention that although LEL points strategy is similar to Use Case Points, our approach can be used at an earlier stage than Use Case Points, since it is applied directly to the LEL and no derivation or writing of Uses Cases is needed.

People who regularly use the LEL will benefit from this approach. Once they have built a LEL, they need to elicit requirements and analyze them, and they may capture this knowledge in the previously built LEL. Then, they can apply our approach with the refined LEL in order to obtain a measure of the application that the LEL describes. Thus, they do not need to produce a requirements specification to apply a traditional technique to estimate its size.

We propose a technique to obtain a measure of the LEL similar to the measure of Unadjusted Use Case Points (UUCP) from the Use Cases. That is, a measure of the functionality of the application. This is possible since the LEL captures the language of the application and some of the expressions captured will be verbs. These verbs are further related to Use Cases [4]. Thus, we claim that the size of the LEL (which we refer to as ULELP) could be used instead of UUCP in the estimation with Use Case Point techniques. We performed a case study to show the applicability of our approach and we carried out an experiment in order to verify its effectiveness. The rest of the paper is organized in the following way. Section 2 presents the background necessary to understand the approach. Section 3 describes the estimation approach. Section 4 provides an example. Section 5 shows a case study. Section 6 presents the evaluation process. Section 7

describes related works. Finally, section 8 brings forward some conclusions and future works.

II. BACKGROUND

This section presents the Language Extended Lexicon, the technique we use to model the application and from which we measure its size. The Use Case Point technique is also presented here, as the approach we propose is based on it.

A. Language Extended Lexicon (LEL)

LEL [21] is a glossary whose goal is to record the definition of terms that belong to a domain. It is tied to a simple idea: *understand the language of a problem, without worrying about the problem.*

Terms (symbols) are defined through two attributes: notion and behavioural responses. Notion describes the intrinsic and substantial characteristics of the symbol (denotation), while behavioral responses (connotation) describe the link between the term being described and others.

There are two principles that must be followed while describing symbols: the circularity principle (also called closure principle) and the minimal vocabulary principle. The circularity principle states that the use of LEL symbols must be maximized when describing a new symbol. The minimal vocabulary principle states that the use of words that are external to the Lexicon must be minimized. These principles are vitally important in order to obtain a self-contained and highly connected LEL. Connections among symbols determine that the LEL can be viewed as a graph.

Each symbol of the LEL belongs to one of four categories: subject, object, verb and state. This categorization guides and assists the requirements engineer with the description of attributes. Table 1 shows each category with its characteristics and how to describe them.

TABLE 1. LEL CATEGORIES.

| Category | Characteristics | Notion | Behavioral responses |
|----------|--|---|---|
| Subject | Active elements which perform actions | Characteristics or condition that subject satisfies | Actions that subject performs |
| Object | Passive elements on which subjects perform actions | Characteristics or attributes that object has | Actions that are performed on object |
| Verb | Actions that subjects perform on objects | Goal that verb pursues | Steps needed to complete the action |
| State | Situations which subjects and objects can be in | Situation represented | Actions that must be performed to change into another state |

Some examples of LEL symbols are presented here. The classic bank application is used to show symbols from each category. The example consists in a bank which allows its clients to open and close accounts. If the account is activated (open) the client can deposit or withdraw money and consult the balance. The bank can also perform a cash audit.

It is important to mention that in the written descriptions we underline the terms which correspond with other defined symbols in order to show the application of the circularity principle. The following examples are: subject *client* in Figure 1; object *account* in Figure 2; verb *withdraw* in Figure 3; and state *activated* in Figure 4.

Subject: client
Notion
Person that operates an account.
Behavioral responses
The client can open an account.
The client can deposit money into his account.
The client can withdraw money from his account.
The client can consult his account balance.
The client can close an account.

Fig. 1. Client symbol description.

Object: account
Notion
The account has a balance.
Behavioral responses
The client can open an account.
The client can deposit money into his account.
The client can withdraw money from his account.
The client can consult his account balance.
The bank performs a cash audit.
The client can close an account.

Fig. 2. Account symbol description.

Verbs: withdraw
Notion
Act of taking money from the account.
Behavioral responses
The bank must check that the account has enough money to perform the withdrawal.
The bank must check that the owner of the account has not withdrawn more times than the limit allows.
The bank must check that the owner of the account doesn't have any credit card debts.
The bank reduces the balance of the account according to the amount withdrawn.

Fig. 3. Withdraw symbol description.

State: Activated
Notion
Situation where the client is ready to use an open account.
Behavioral responses
The client can close the account and he will have a closed account.

Fig. 4. Activated symbol description

B. Use Case Points (UCP)

The Use Case Point method is a software sizing and measurement that uses Use Case Documents and is based on [18]. This work is an adaptation of that done by Allen Albrecht on function points [2].

UCP states that the time to construct the application is affected by:

- (i) The number and complexity of Use Cases.
- (ii) The number and complexity of actors.
- (iii) The technical requirements of the application, such as concurrency, security and performance.
- (iv) Various environmental factors such as the development team's experience and knowledge.

UCP analyzes Use Case scenarios, actors and various technical and environmental factors:

- (i) Unadjusted Use Case Points (UUCP). This value includes complexity of the Use Cases and the actors.
- (ii) Productivity Factor (PF).
- (iii) Technical Complexity Factor (TCP).
- (iv) Environment Complexity Factor (ECF).

All these factors are combined in the following equation:

$$UCP = UUCP * PF * TCP * ECF$$

Fig. 5. Use Case Points equation.

1) *Unadjusted Use Case Points (UUCP)*. Unadjusted Use Case Points are calculated based on two values:

- (i) The Unadjusted Use Case Weight (UUCW) based on the classes, database entities, and the total number of activities (or steps) contained in all the Use Case Scenarios.
- (ii) The Unadjusted Actor Weight (UAW) based on the combined complexity of all the Use Cases Actors.

2) *Unadjusted Use Case Weight (UUCW)*. Individual use cases are categorized as Simple, Average or Complex, and weighed mainly depending on the classes, database entities, and the number of steps they contain. Table 2 summarizes the characteristics.

TABLE 2. USE CASES RANKS.

| Use Case Type | Description | Weight |
|---------------|--|--------|
| Simple | It is a simple user interface and accesses only a single database entity; its success scenario has 3 steps or less; its implementation involves less than 5 classes. | 5 |
| Average | It involves more interface design and accesses 2 database entities; between 4 and 7 steps; its implementation involves between 5 and 10 classes. | 10 |
| Complex | It involves a complex user interface or processing and accesses 3 or more database entities; over seven steps; its implementation involves more than 10 classes. | 15 |

3) *Unadjusted Actor Weight (UAW)*. Actors are classified as Simple, Average or Complex in relation to the external interface they represent in the software system. Human

interacting through graphical user interface represents the Complex level, while another software system which interacts through an API is the Simple rank. Table 3 summarizes the characteristics of the actors.

TABLE 3. ACTOR RANKS.

| Actor Type | Description | Weight |
|------------|---|--------|
| Simple | Another system through an API. | 1 |
| Average | Another system through a protocol. A person through a text-based user interface | 2 |
| Complex | A person through a graphical user interface | 3 |

4) *Productivity Factor*. The Productivity Factor (PF) is a ratio of the number of man hours per Use Case Point based on past projects. If no historical data has been collected, a figure between 15 and 30 is suggested by industry experts. A typical value is 20.

5) *Technical Complexity Factors*. Thirteen standard technical factors exist to estimate the impact on productivity that various technical issues have in an application. Each factor is weighed according to its relative impact. A weight of 0 indicates that the factor is irrelevant, while value 5 means that the factor has the most impact.

6) *Environmental Complexity Factors*. Eight environmental complexity factors are defined and they must be weighed in a similar way as technical complexity factors. According to Ribu [26], environmental factors play a very important role in the estimation. A slight variation will increase the Use Case Point by a very drastic amount.

III. OUR APPROACH

The counting scheme proposed must be applied to a LEL which describes symbols of a specific application. It is important to mention that the LEL must not describe an application domain language which can give origin to several different applications; instead, the LEL must describe a specific application. Thus, before performing the counting scheme, the application domain LEL must be refined into a specific application LEL, for example by removing the verb symbols which will not be developed as functionality.

The counting scheme proposed is based on the Use Case Points estimation technique. Since Use Cases can be derived from the LEL [4], we designed a counting scheme that uses the same calculations performed in Use Case Points, but the calculations are performed in the LEL using the relationship in the information from both models. It is worth mentioning that the measure calculated with this approach is equivalent to Unadjusted Use Case Points (UUCP), which we refer to as Unadjusted LEL Points (ULELP). This measure includes Unadjusted Verbs Weight (UVW) and Unadjusted Subject Weight (USW) which are equivalent to Unadjusted Use Case Weight (UUCW) and Unadjusted Actor Weight (UAW). Our approach does not consider measures

equivalent to: (i) Productivity Factor, (ii) Technical Complexity Factors and (iii) Environmental Complexity Factors. Since ULELP is based on UUCP and considering that (i), (ii) and (iii) do not depend on functionality described by Use Cases, occasionally the UCP framework for (i), (ii) and (iii) can be used with ULELP. The rest of the section describes how to calculate USW and UVW.

A. Unadjusted Subject Weight (USW)

The measure of UAW in UCP consists in analyzing the actors who use the application. These actors are modeled as symbols of the subject category in LEL descriptions according to [4]. Thus, subject symbols must be ranked in a similar way as actors of UCP are ranked.

TABLE 4. SUBJECT RANKS.

| Subject Type | Description | Weight |
|--------------|---|--------|
| Simple | Another system through an API. | 1 |
| Average | Another system through a protocol. A person through a text-based user interface | 2 |
| Complex | A person through a graphical user interface | 3 |

B. Unadjusted Verb Weight (UVW)

The measurement of UUCW in UCP consists in analyzing the following characteristics for each piece of functionality (UC): (i) user interface, (ii) access to database entity, (iii) number of steps in the success scenario and (iv) number of classes that involve its implementation. Most of this information is also present in the LEL according to [4].

Although a LEL models the language of an application, we must consider that the language corresponds to a specific application, thus we must analyze the user interface in the same way as it is analyzed in UCP.

The UC has a description of the main success scenario. In verb symbols, this description is given in their behavioural responses. Therefore, the number of steps is counted according to the number of steps in behavioural responses. Database entities and classes must be analyzed from this attribute.

Since a LEL describes the language of the application, it does not explicitly mention database entities or classes. Nevertheless, the LEL considers symbols of the object and subject categories, which are related to database entities and classes. Wirfs-Brock et al. [30] also make this connection when they relate nouns to objects. Then, in an Object Oriented application, some objects will be persisted into a database, thus, they can be considered database entities.

In general, every object symbol of the LEL can be considered a database entity in UCP. But as the LEL is very detailed, it can have descriptions of object symbols which are not commonly described in UC, and they will only appear in a detailed Entity Relationship model as attributes instead of entities. Thus, symbols of the object category and with a low level of detail must not be considered.

Additionally, symbols of the subject category which are used passively must be taken into account. Although they are subjects, if they are used passively as objects they must be considered objects too.

The last element to count in UC is the number of classes that involve its implementation. In the LEL description, subject symbols are candidates to be implemented as classes because they are active elements which perform actions (this is the definition of a class). Nevertheless, we must also consider objects and count them when they perform actions too.

Finally, Table 5 describes how to rank verb symbols.

TABLE 5. VERBS RANK.

| Verb Type | UI | Objects | Steps | Subjects | Weight |
|-----------|---------|---------|-------|----------|--------|
| Simple | Simple | 1 | <=3 | <=4 | 5 |
| Average | Average | 2 | 4-7 | 5-10 | 10 |
| Complex | Complex | >2 | >7 | >10 | 15 |

IV. AN EXAMPLE

In this section we show how to apply the strategy proposed. We use an example from a small application extracted from a Human Resources Management system. We describe how to rank and measure three verb symbols in order to exemplify the strategy. Although the example is very small, it depicts the three situations that can be encountered when applying the counting strategy.

The rest of the section is organized in the following way. First, a description of the application is given. Then, the symbols of the LEL identified from the application are shown. Finally, subject and verb symbols are described and analyzed according to the strategy proposed so as to rank them.

A. The Application

The software application's goal is to manage vacation periods for the employees of a company. Employees have a certain number of vacation days per year. This number depends on how many years the employee has been working for the company. When the employee requests a vacation period, the request needs two authorizations to become effective. First of all, the Human Resources Department must verify that the employee is not requesting more days than those he is allowed to take. After that, the employee's manager must analyze the project schedule in order to determine whether or not the employee's presence is essential at work in the vacation period he has requested.

B. LEL

The symbols identified from the vacation management application are summarized in Table 6.

TABLE 6. LEL SYMBOLS OF VACATION APPLICATION.

| Subjects | Objects | Verbs | States |
|----------------------------|------------------|------------------|----------|
| Employee | Vacation Request | Request vacation | New |
| Human Resources Department | Period | Verify request | Verified |
| Manager | | Analyze request | Analyzed |
| | | | Approved |
| | | | Rejected |

C. Ranking of Elements

ULELP is calculated from USW and UVW. There are 3 subjects in the example: *employee*, *human resources department*, and *manager*. We can consider that the application will have a graphical user interface. Thus, each subject is considered complex and its weight is 3. So, there are 3 subjects with a weight of 3 each. Therefore, USW is 9.

In the following paragraphs we describe each verb symbol and also analyze steps, objects and subjects in order to rank them. In general, all the user interfaces are simple, so they are not mentioned. The first verb symbol is *request vacation*, which is described in Figure 6.

| |
|---|
| Verb: request vacation |
| Notion |
| Act of asking for permission to take some days off work |
| Behavioral responses |
| The <u>employee</u> defines the time <u>period</u> |
| Human Resources Department records the vacation request |

Fig. 6. Request vacation symbol description.

The strategy demands to analyze three variables in order to rank the symbol: (i) number of steps, (ii) number of objects and (iii) number of subjects. All of them are related to behavioural responses. The number of steps in the behavioral responses is two, shown in each sentence. Although the number of objects in the behavioural responses is also two (*period* and *vacation request*), we must consider only one object: *vacation request*, because *period* would actually be an attribute of *vacation request*, and would therefore not be an object, so it must not be taken into account. Finally, the number of subjects is two: *employee* and *Human Resources Department*. According to these variables, the verb symbol *request vacation* must be ranked as a simple verb. Table 7 summarizes this information.

TABLE 7. RANKING OF THE REQUEST VACATION SYMBOL

| Verb Type | UI | Objects | Steps | Subjects | Weight |
|-----------|--------|---------|-------|----------|--------|
| Simple | Simple | 1 | 2 | 2 | 5 |

The following verb symbol to analyze is *verify request*, which is described in the Figure 7.

The number of steps is three, because there are three sentences in the behavioral responses. The number of objects in the behavioral responses is one (vacation request), but we must count one more object. Although employee is a subject symbol, it is used as an object, so two objects must be counted. Finally, the number of subjects is two: Human Resources Department and employee. Although three variables are in simple rank, there is one variable in average rank.

| |
|---|
| Verb: verify request |
| Notion |
| Act of verifying that the number of days requested by the <u>employee</u> does not exceed the days allowed to him. |
| Behavioral responses |
| <u>Human Resources Department</u> identifies the number of days that the <u>employee</u> can take. |
| <u>Human Resources Department</u> calculates the number of days that the <u>employee</u> has already taken, if any. |
| <u>Human Resources Department</u> checks that the number of days in his <u>vacation request</u> do not exceed the number of remaining days allowed. |

Fig. 7. Verify request symbol description.

Using the same criteria as with UCP, the verb symbol: *verify request* must be ranked as an average verb because one variable (objects) is ranked as average. Table 8 summarizes this information.

TABLE 8. RANKING OF THE VERIFY REQUEST SYMBOL

| Verb Type | UI | Objects | Steps | Subjects | Weight |
|-----------|--------|---------|-------|----------|--------|
| Simple | Simple | | 3 | 2 | |
| Average | | 2 | | | 10 |

The last verb symbol to analyze is: *analyze request*. It is described in Figure 8. The number of steps in behavioural responses is one. Although the manager needs to do some comparison with the project schedule, the contrast he must do is beyond the scope of the application; this is the reason why the symbol *contrast* is not defined and there are no more steps in describing the constraint. The number of objects is one: *vacation request*. The *schedule tasks* are mentioned, but they are beyond the scope of the application, too. Finally, the number of subjects is one: *Manager*. According to these variables, the verb symbol *analyze request* must be ranked as a simple verb. It is important to mention that the application only needs to make it possible for the *manager* to accept or reject the *vacation request*. This functionality is very easy to implement, so ranking it as simple makes sense. Table 9 summarizes this information.

| |
|---|
| Verb: analyze request |
| Notion |
| Act of analyzing the project schedule needs for the period requested. |
| Behavioral responses |
| <u>Manager</u> contrasts the <u>vacation request</u> with the schedule tasks. |

Fig. 8. Analyze request symbol description.

TABLE 9. RANKING OF THE ANALYZE REQUEST SYMBOL

| Verb Type | UI | Objects | Steps | Subjects | Weight |
|-----------|--------|---------|-------|----------|--------|
| Simple | Simple | 1 | 1 | 1 | 5 |

Summing up, USW is $3+3+3$, and UVW is $5+10+5$. Thus, ULELP is USW + UVW, that is, 29. This value could be used instead of UUCP in UCP. Adding the Technical Complexity Factor (TCF), the Environment Complexity Factor (ECF) and the Productivity Factor (PF), an effort estimation in man hours could be reached.

V. CASE STUDY

In order to verify and show the applicability of the approach, we applied the counting strategy to a real software system. The system is an open government platform which integrates different tools: a social network, a content repository, instant messaging, voice over IP communication, and many others. Some of the tools were developed by us while others were open source software. The development began in 2010 and nowadays there are 22 team members who play different roles: leaders, architects, analysts, testers, UX designers, and developers.

We built the LEL with the knowledge gathered from being part of the team and we also consulted a document in order to validate the completeness of the LEL. The vision and scope document was used. This document describes the boundaries and features of the system, but it does not describe the requirements. It is worth mentioning that we use this document because the focus of the case study is to verify the counting strategy and not to assure completeness of the LEL. The analyst needs to elicit and analyze requirements to refine the LEL. This activity can be done either interacting with a stakeholder or using the vision and scope document as we did for this case study. Moreover, we need a LEL and Use Cases that represent the same system in order to compare their measures, thus we need to ensure that both models represent the same functionality.

We identified a total of 234 symbols. The number of symbols from each category is shown in Table 10.

All of the subjects correspond to people using the system through a graphical user interface. Thus, they are ranked as complex. Each of the 8 subjects has a weight of 3, so USW is $8 * 3 = 24$. Then, verbs are ranked according to Table 11.

TABLE 10. NUMBER OF SYMBOLS FOR EACH CATEGORY OF THE SOCIAL NETWORK SYSTEM

| Category | Symbols |
|----------|---------|
| Subjects | 8 |
| Objects | 26 |
| Verbs | 170 |
| States | 30 |

TABLE 11. VERBS RANK.

| Verb Type | Number |
|-----------|--------|
| Simple | 140 |
| Average | 23 |
| Complex | 7 |

Thus, UVW is $140 * 5 + 23 * 10 + 7 * 15 = 1035$. Finally, the Unadjusted LEL Points (ULELP) is USW + UVW = 1059. This case study allows us to verify the applicability of the approach and the proximity between the measure of ULELP and UUCP. The strategy to measure USV is quite simple and similar to UAW in UCP. We identified 8 subject symbols which match the 8 actors identified in the Use Case analysis. Thus, considering that the application has a graphical user interface, USW was 24 (equal to the UAW).

The weight of verbs is different from the weight of Use Cases. The main difference arises from the functionality related with workflows. Each transition of the workflow was modeled as an individual verb symbol, while one Use Case describes functionality related to several steps in the workflows. Thus, 12 verbs were ranked as simple while the same functionality was represented by 4 Use Cases ranked as average. There were also 10 verb symbols ranked as simple while their related functionality was modeled into 2 Use Cases ranked as complex. This was a remarkable difference in the definition of the LEL and the Use Cases. The difference had an impact on the measurement, but it was small. The weight related to these symbols was 110, while the weight related to the Use Cases was 70.

There were other verb symbols which were ranked as a different category from their related Use Cases. These 11 verb symbols were ranked as simple while their related Use Cases were ranked as complex. The reason for the different ranks lies in the description of the symbols, as they were not fully described; that is, their related Use Cases have more information than the LEL symbols.

Table 12 summarizes the Use Case ranks.

TABLE 12. USE CASES RANK.

| Use Case Type | Number |
|---------------|--------|
| Simple | 107 |
| Average | 28 |
| Complex | 19 |

Thus, UUCW is $107 * 5 + 28 * 10 + 19 * 15 = 1100$. Finally, the Unadjusted Use Case Points value (UUCP) is $UAW + UUCW = 1124$.

In summary, the ULELP and UUCP measures were similar. In fact, they were quite close: 1059 and 1100 respectively. The analysis of UAW and USW is similar in both approaches, so it is not necessary to perform any comparison between them. The analysis of UUCW and UVW depends on the construction of each model. The description of functionality related to workflows is modeled in a different way in the LEL and UC. Although the weight of both models was quite close too, we must continue to consider and study this situation.

VI. EVALUATION

An experiment was performed in order to assess the precision of the strategy proposed. Since our approach is based on Use Case Points, the experiment consisted in comparing our strategy with Use Case Points. The previous case study showed that the difference between Use Case Points and LEL Points lies in the Unadjusted Use Case Weight and the Unadjusted Verb Weight. Unadjusted Actor Weight and Unadjusted Subject Weight were equal in both strategies, thus we focused on contrasting UUCW with UVW in the experiment. There are four attributes that define UUCW and UVW: (i) user interface (UI), (ii) database entities / objects, (iii) steps, and (iv) classes / subjects. Since the analysis of UI is similar in both approaches, we did not consider this attribute in the experiment. We decided to design an experiment where we provided the participants with a description of Use Cases and LEL, so we could measure the subjectivity in the application of the strategy with the same model. It is irrelevant to compare steps between Use Cases and LEL, because this measure is objective. We were particularly interested in comparing how subjective the identification of the other two variables is in the two strategies. The goal of the experiment is described according to the Goal/Question/Metric (GQM) method formulated by Basili et al. [7]:

Analyze Unadjusted Use Case Weight and Unadjusted Verb Weight for the purpose of contrasting the variables “database entities vs object” and “Classes vs Subjects” with respect to precision

The participants of the experiment were 18 students from a Computer Science postgraduate course. The experiment was part of the course activities. All the students had a degree in Computer Science and experience in the software development industry, some of them as developers, others as analysts and some others as team leaders. Some participants were also teachers at the University. The majority of the participants were Argentinean, but there were also people from Colombia.

The materials used consisted of a slide show presentation, a basic guide about how to apply Use Case Points and LEL

Points, and a description of the application, which was an issue tracker system. Three different materials were produced: (i) a colloquial description, (ii) Use Cases and (iii) a LEL. From the same colloquial description, 10 Use Cases and their related verb symbols were described. This functionality was: (i) create an issue, (ii) assign an issue, (iii) start, (iv) pause, (v) finish and (vi) cancel working on an issue, (vii) break down an issue, (viii) browse issues, (ix) calculate working factor for each member (in order to perform resource leveling) [25], and (x) perform resource leveling.

The experiment was carried out during a class. At the beginning of the experiment, the participants were instructed how to calculate UUCP and ULELP (all the participants were instructed in both strategies). Then, two groups were formed. A group of 9 students was asked to apply Use Case Points while another group of 9 students was asked to apply LEL points. We decided to make two groups of people and ask them to apply only one strategy each, in order to avoid bias from one technique to the other. The participants were asked to complete the counting in two hours. First, they needed to read the description of the application, because they were not given any information about it during the presentation, in which only the counting strategies were described. After that, every participant had to provide the following information for each Use Case or Verb: (i) Database entities or Objects identified and (ii) Classes or Subjects identified.

The analysis consisted in calculating the variance for each attribute in each technique for every requirement. These variances allow us to determine the precision of the techniques. The number of participants was small in order to obtain statistical results. Nevertheless, the experiment allows us to obtain general qualitative information to make adjustments and carry out the experiment again with more participants. For this reason, instead of showing the values for each variance, we categorize situations into four groups according to the difference in variance:

(A) Both variances of ULELP and UUCP are low and close to each other. Both techniques are equally good.

(B) The variance of ULELP is low; the variance of UUCP is high. ULELP is more precise than UUCP.

(C) The variance of ULELP is high; the variance of UUCP is higher. Neither approach is precise, but this high variance does not impact on the ranking of both strategies.

(D) The variance of ULELP is high; the variance of UUCP is higher. Neither approach is precise, but this high variance impacts on the ranking of both strategies.

In relation to database entities / object comparison, both techniques were equally good in 4 out of 10 requirements. Features i (create an issue) and vii (break down an issue) were more precisely ranked with ULELP than with UUCP. Features viii (browse issues), ix (calculate working factor) and x (perform resource leveling) were not precise in either technique.

TABLE 13. COMPARISON OF THE VARIANCE OF EACH VARIABLE IN USE CASE AND LEL FOR ALL REQUIREMENTS

| Req | Variance of Database Entity in UUCP | Variance of Objects in ULELP | Variance of Classes in UUCP | Variance of Subjects in ULELP |
|------|-------------------------------------|------------------------------|-----------------------------|-------------------------------|
| i | B | B | A | A |
| ii | D | D | A | A |
| iii | A | A | A | A |
| iv | A | A | A | A |
| v | A | A | A | A |
| vi | A | A | A | A |
| vii | B | B | A | A |
| viii | D | D | A | A |
| ix | D | D | C | C |
| x | D | D | C | C |

In relation to class / subject comparison, both techniques were equally good in 8 out of 10 requirements. Features ix (calculate working factor) and x (perform resource leveling) were not precise in either technique, but as the ranks are 0-4, 5-10 and 11-more this difference did not impact on rank definition.

In summary, the variance of ULELP was equal to or better than UUCP. We can thus claim that ULELP is more precise than UUCP. The reason for this is that participants must count objects and subjects and they are assisted in this by LEL symbols, while they do not have this help when they rank with UUCP. This experience allows us to design a new experiment with more participants in order to prove this finding statistically. Moreover, the new experiment will include the description of Use Cases and LEL by participants, so as to assess the variance in number of steps.

Feldt et al [13] state the importance of analyzing threats to the validity of the study and the results. Wohlin et al. [31] group validity threats into four categories: conclusion, internal, construct and external validity. The following paragraphs analyze different threats from each category.

Concerning conclusion category, one possible threat is random heterogeneity of subjects. The participants are heterogeneous as regards years of experience in industry, but there is homogeneity regarding overall experience. Since the application of the technique is very simple, heterogeneity of subjects does not represent any threat.

The second category of threats to analyze is internal validity. Selection is the main threat to internal validity. In order to tackle the effect of natural variation in human performance we selected people with experience in the application domain and in requirements engineering, but with no experience in the approaches we analyzed. Then, we carried out a randomization to assign subjects to treatments.

According to the construct validity category, we observed that the experiment did not suffer from such threats referred to as hypothesis guessing, evaluation apprehension or experimenter expectancies. The subjects were not familiar with the approaches, so they could not force specific results. Also, the experiment did not have to deal with the interaction of different treatments, because each subject was assigned only one treatment, so there was no bias.

Sjøberg et al [27] state that many threats to external validity are caused by an artificial setting of the experiment. Taking this into account, we set up an experiment which had the complexity of a small but real application (a real application was developed in an organization where one of the authors works).

VII. RELATED WORKS

The goal of our approach is to measure the size of an application. Niknafs [23] states that while a requirements engineer has in-depth domain knowledge that helps him to understand the problem easier, he can nevertheless fall for tacit assumptions of the domain and might overlook issues that are obvious to domain experts. Since we model the application through a glossary that must be refined from a general glossary of the domain, our approach makes it possible to tackle the problem stated by Niknafs, since one requirements engineer can write the general glossary, while another can write the specific application glossary.

Our approach is also in agreement with Glinz [14] and Waldmann [29]. Glinz proposes a lightweight requirements modeling language as an alternative to textual and pictorial specifications. Waldmann states that requirements engineering must learn from agile development. In our approach, requirements specifications can be automated from the LEL as in [8].

LEL can be considered a lightweight requirements model to be used in agile methodologies. Although a LEL is an early product, the measurement from early products (and early activities) was shown to be effective by Tsunoda in [28]. This early measurement is also important for business decisions [20], and it is crucial for small companies that do not have a defined process [6].

Abrahao [1] proposes a measurement procedure (ReqPoints) to estimate the size of object-oriented software projects from a requirements specification. Specifically, a set of measurement rules is defined as a mapping between the concepts of the Requirements Metamodel onto the concepts of the Function Point Analysis (FPA) Metamodel. Our approach is similar to Abrahao's since the elementary functions identified are similar to verb symbols. Then, they map the elements onto Function Points while we map them onto Use Case Points.

Harpur et al [16] present an approach to apply FPA to an object-oriented requirements model which is specified with scenarios as well as sequence diagrams and class diagrams. This approach defines rules to interpret the object-oriented model and apply FPA. It is similar to our approach because in some way we provide a strategy to interpret the LEL as if it were UC; nevertheless our translation is simpler. This is an important distinction: we work with a simple and easy to use technique, i.e., LEL. These features are essential to foster applicability and improve the obtained results. Anda et al [3] and Ribu [26] report the difficulty of estimating from complex structures. Anda reports the results of three

industrial case studies on the application of a method based on Use Case Points. They state that the design of the use case models has a strong impact on the estimation, since the more complex the design of the product, the more difficult and more sensible the application of the estimation techniques is. Ribu agrees with these views, reporting that the main difficulty she encountered when applying the Use Case Points method was that practitioners wrote use cases in very different ways and with different levels of detail. Our approach relies on the LEL, and although different subjects can bias the descriptions, LEL has very basic rules which help to write homogeneous descriptions.

Cockcroft [11] empirically proved that it is possible to calculate the size of a data flow diagram (which is built early at software development) and this size is related to the code line of the coded application. MacDonell [22] works on measuring the specification and the relationship between specification and process effort. We work with the specification of the application, but the size we obtain can be translated to effort because there is a relationship between LEL-points and Use Case Points obtained from the Use Cases derived from a LEL. Zhao [32] calculates a measuring from an ER diagram. He developed a complexity path in a data-oriented model. Although our model is functional oriented, it also has object symbols which make a similar analysis possible. Grimstad [15] states that estimation-irrelevant information should be removed from the requirements specification prior to its use as the input to estimate work. Since we use raw material to construct Use Cases, our approach is in agreement with Grimstad's.

VIII. CONCLUSION

We have presented an approach to calculate the size of an application from the application language captured by the Language Extended Lexicon. Estimating the size of the application prior to requirements specification can be very difficult, and wrong measures are dangerous for project planning. With the proposed approach we focus on the language, and from there we estimate the size of the application. With this size in mind, a more realistic estimation can be performed before the existence of a requirements specification.

Our approach consists in analyzing a Language Extended Lexicon in the same way as Use Case Points analyzes Use Cases. Since there is a strategy to derive Use Cases from a Language Extended Lexicon, we have adapted Use Case Points based on Use Case derivation from LEL in order to measure a LEL. Thus, if we have a LEL, we can measure it directly and avoid the extra effort of deriving Use Cases and then calculating Use Case Points from such a derivation. We obtain a similar measure with less effort because our technique works with a previous product. Moreover, we obtain a measure earlier. Although it is necessary to elicit and analyze requirements to transform the domain LEL into a specific application LEL, we believe it is useful for

requirements engineers who use LEL, as the extra effort they make adjusting the LEL is less than the effort necessary to specify Use Cases. From that LEL, Unadjusted LEL Points can be semi-automatically calculated obtaining a measure of the system previous to specifying its requirements.

The similarities between Use Case Points and LEL Points arise from the fact that the design of LEL Points is based on Use Case Points. In this paper we described an experiment which showed that the measures obtained using LEL Points have an equal or smaller variance than Use Case Points, thus people who do not use LEL regularly, can begin using it in order to get benefits from the better precision our technique has. In order to further sustain this initial result, we are working on more evaluations based on different case studies. Particularly, we are working on obtaining measures from enough participants in order to perform a statistical analysis. Moreover, we will work in analyzing relationships such as “is a” between symbols and their impacts on measurement. We are also working in tool support to partial automate the counting process.

REFERENCES

- [1] S. Abrahao, and E. Insfran, “A metamodeling approach to estimate software size from requirements specifications”, in Proceeding of the Software Engineering and Advanced Applications (SEAA'08), ISBN 978-0-7695-3276-9, 3-5 Sept, Parma, 2008, pp 465 – 475.
- [2] A. J. Albrecht, “Measuring application development productivity”, in Proc. IBM Applications Development Symp., GUIDE Int. and SHARE Inc., IBM Corp., Monterey, CA, 1979, p. 83.
- [3] B. Anda, H. Dreiem, D. I. K. Sjøberg, and M. Jørgensen, “Estimating software development effort based on Use Cases – Experiences from industry”, in Proc. of the 4th Int. Conference on the Unified Modeling Language, Springer Verlag, Toronto, Canada, 2001, pp. 487-502.
- [4] L. Antonelli, G. Rossi, J.C.S.P. Leite, and A. Oliveros, “Deriving requirements specifications from the application domain language captured by Language Extended Lexicon”, in proceedings of the Workshop in Requirements Engineering (WER), Buenos Aires, Argentina, April 2012, pp 24 – 27.
- [5] L. Antonelli, G. Rossi, J.C.S.P. Leite, and J. Araújo, “Early identification of crosscutting concerns with the Language Extended Lexicon”, Requirements Engineering Journal, <http://dx.doi.org/10.1007/s00766-013-0193-4>, Springer London, 2013, pp 1-23.
- [6] J. Aranda, S. Easterbrook, and G. Wilson, “Requirements in the wild: how small companies do it”, in Proc of the 15th IEEE International Requirements Engineering Conference (RE '07), ISBN 978-0-7695-2935-6, 10.1109/RE.2007.54, 15-19 Oct. 2007, Delhi, 2007, pp 39 – 48.
- [7] V.R. Basili, G. Caldiera, and H.D. Rombach, “The goal question metric approach”, in Encyclopedia of Software Engineering, John Wiley & Sons, Vol. 1, 1994, pp.528-532.
- [8] E. Boutkova and F. Houdek, “Semi-automatic identification of features in requirement specifications”, in Proceedings of the 19th IEEE International Requirements Engineering

- Conference (RE), ISBN 978-1-4577-0921-0, 10.1109/RE.2011.6051626, Aug. 29 2011-Sept. 2, Trento, 2011, pp 313 - 318.
- [9] K.K. Breitmann and J.C.S.P. Leite, "Ontology as a requirements engineering product", in Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE), IEEE Computer Society, Monterey Bay, California, USA, ISBN 0-7695-1980-6, 2003.
- [10] E.R. Carroll, "Estimating software based on use case points", Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, San Diego, CA, USA, October 16-20 2005.
- [11] S. Cockcroft, "Estimating CASE development size from outline specifications", Information and Software Technology 38, 1996, pp. 391-399.
- [12] Cosmic Common Software Measurement International Consortium, Cosmic Functional Size Measurement. Available at: <http://www.cosmicon.com/>.
- [13] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research - An initial survey", in proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE), July 1-3, San Francisco Bay, 2010, pp 374-379.
- [14] M. Glinz, "Very lightweight requirements modeling", in Proceeding of the 18th IEEE International Requirements Engineering Conference (RE), ISBN 978-1-4244-8022-7, 10.1109/RE.2010.73, Sept. 27 2010-Oct. 1, Sydney, NSW, 2010, pp 385 – 386.
- [15] S. Grimstad and M. Jorgensen, "The impact of irrelevant information on estimates of software development effort", in Proceedings of the 18th Australian Software Engineering Conference (ASWEC 2007), ISBN 0-7695-2778-7, 10.1109/ASWEC.2007.48, 10-13 April, Melbourne, Vic., 2007, pp 359 – 368.
- [16] V. Harput, H. Kaindl, and S. Kramer, "Extending Function Point analysis to Object-Oriented requirements specifications", in Proceeding of the 11th IEEE International Symposium of Software Metrics, September 2005.
- [17] International Function Points User Groups, available at: <http://www.ifpug.org/>.
- [18] G. Karner, "Metrics for objectory", Master Thesis, Linkoping University (LiTH-IDA- Ex-9344:21), Linkoping, Sweden, 1993.
- [19] S. Kusumoto, M. Fumikazu, H. Shigeo, and M. Yuusuke, "Estimating effort by Use Case Points: method, tool and sase study", Proceedings of the 10th International Symposium on Software Metrics, 2004.
- [20] L. Lehtola, M. Kauppinen, and J. Vähäniitty, "Strengthening the link between business decisions and RE: Long-term product planning in software product companies", in Proceedings of the 15th IEEE International Requirements Engineering Conference (RE '07), ISBN 978-0-7695-2935-6, 10.1109/RE.2007.30, 15-19 Oct., Delhi, 2007, pp 153 – 162.
- [21] J.C.S.P. Leite and A.P.M. Franco, "A strategy for conceptual model acquisition", in Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, California, IEEE Computer Society Press, 1993, pp 243-246.
- [22] S.G. MacDonell, "Establishing relationships between specification size and software process effort in CASE environments", Information and Software Technology, 39 - 1, 1997, pp 35-45.
- [23] A. Niknafs and D.M. Berry, "The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation", in Proceeding of the 20th IEEE International Requirements Engineering Conference (RE), 10.1109/RE.2012.6345802, ISBN 978-1-4673-2783-1, 24-28 Sept, Chicago, IL, 2012, pp 181 – 190.
- [24] J. Ouwerkerk and A. Abran, "Evaluation of the design of Use Case Points (UCP)", MENSURA2006, Shaker Verlag, 4-5 November, Cadiz, Spain, 2006.
- [25] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK Guide), Fifth Edition, ISBN 9781935589679, 2013.
- [26] K. Ribu, "Estimating Object-Oriented software projects with Use Cases", Master of Science Thesis, University of Oslo Department of Informatics, 7th November 2001.
- [27] D.I.K. Sjøberg, B. Anda, E. Arisholm, T. Dybå, M. Jørgensen, A. Karahasanovic, E.F. Koren, and M. Vokác, "Conducting realistic experiments in software engineering", in Proceedings of the International Symposium on Empirical Software Engineering (ISESE), ISBN 0-7695-1796-X, 2002, pp 17.
- [28] M. Tsunoda, Y. Kamei, K. Toda, and M. Nagappan, "Revisiting software development effort estimation based on early phase development activities", in Proceeding of the 10th IEEE Working Conference on Mining Software Repositories (MSR), ISBN 978-1-4799-0345-0, 10.1109/MSR.2013.6624059, 18-19 May, San Francisco, CA, 2013, pp 429 - 438.
- [29] B. Waldmann and A.G. Phonak, "There's never enough time: Doing requirements under resource constraints, and what requirements engineering can learn from agile development", in Proceedings of the 19th IEEE International Requirements Engineering Conference (RE), ISBN 978-1-4577-0921-0, 10.1109/RE.2011.6051626, Aug. 29 2011-Sept. 2, Trento, 2011, pp 301 – 305.
- [30] R. Wirfs-Brock, B. Wilkerson, and L. Wiener, "Designing Object-Oriented software", Prentice Hall, 1990.
- [31] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, "Experimentation in software engineering an introduction", ISBN 0-7923-8682-5 Academic Publishers 2000.
- [32] Y. Zhao, H.B. Kuan Tanm, and W. Zhang, "Software cost estimation through conceptual requirement", in Proceedings of the Third International Conference on Quality Software, ISBN 0-7695-2015-4, 10.1109/QSIC.2003.1319096, 6-7 Nov. 2003, pp 141 - 144.

The Role of Legal Expertise in Interpretation of Legal Requirements and Definitions

David G. Gordon
Engineering and Public Policy
Carnegie Mellon University
Pittsburgh, USA
dggordon@cmu.edu

Travis D. Breaux
Institute for Software Engineering
Carnegie Mellon University
Pittsburgh, USA
breaux@cs.cmu.edu

Abstract—Government laws and regulations increasingly place requirements on software systems. Ideally, experts trained in law will analyze and interpret legal texts to inform the software requirements process. However, in small companies and development teams with short launch cycles, individuals with little or no legal training will be responsible for compliance. Two specific challenges commonly faced by non-experts are deciding if their system is covered by a law, and then deciding whether two legal requirements are similar or different. In this study, we assess the ability of laypersons, technical professionals, and legal experts to judge the similarity between legal coverage conditions and requirements. In so doing, we discovered that legal experts achieved higher rates of consensus more frequently than technical professionals or laypersons and that all groups had slightly greater agreement when judging coverage conditions than requirements, measured by Fleiss' K . When comparing judgments between groups using a consensus-based Cohen's Kappa, we found that technical professionals and legal experts exhibited consistently greater agreement than that found between laypersons and legal experts, and that each group tended towards different justifications, such as laypersons and technical professionals tendency towards categorizing different coverage conditions or requirements as equivalent if they believed them to possess the same underlying intent.

Index Terms—legal requirements, compliance, statutory interpretation, legal coverage

I. INTRODUCTION

National and provincial laws are enacted in an attempt to address growing privacy and security concerns by restricting how and when personal data may be obtained, used, and protected. This may include obtaining consent from an individual prior to collecting data or after organizational policies have changed, encrypting or otherwise securing data in storage or in transit, and restricting the roles of parties with whom that information can be shared. Laws, including statutes, directives, and regulations, issued in the past five years include numerous U.S. data breach notification laws, updates to the Health Insurance Portability and Accountability Act (HIPAA), and revisions to the European Data Protection Directive, among others [2].

The cost and complexity of compliance with these laws – particularly in multi-jurisdictional contexts – is nontrivial [8, 31] and can have profound effects on system design and product strategy, such as influencing the attractiveness of a new

market or the viability of collecting, storing, and securing data from certain jurisdictions or with certain resources, e.g. an overseas cloud storage service [19]. When addressing the issue of compliance with a new or updated law, an organization must interpret the text and determine whether or not it is covered by the law (i.e. does the law apply to the organization) and consequently what actions are permitted, required, or prohibited to achieve compliance, which we define as alignment of the organization's goals with those prescribed through legislation.

In large companies, these determinations are relegated to a legal department with attorneys and paralegals whose additional duties include discovering recently enacted laws, assessing the risk of noncompliance, and offering consultative advice to IT projects as needed. However, legal resources – be they in the form of an internal legal department, or external firm – may not be utilized in all applicable cases. For example, a 2012 survey of 352 mobile application developers found that 75% of mobile application companies have five or fewer employees, with 40% of those companies consisting of a single employee: the developer of the mobile application [10]. In this case, the mere presence of legal expertise is doubtful, and the organization may be unwilling, or financially unable, to retain legal advice. Even if the firm can afford legal advice, it would be desirable to guide the legal expert to assess specific technical problems relevant to their mobile market segment. In these scenarios, interpretation of the text may be consigned to individuals with little to no legal expertise.

In this paper, we investigate the within-group and between-group similarity with which individuals with no legal background, hereafter referred to as laypersons, technical professionals, and legal experts interpret legal texts. To answer this question, we conducted an empirical survey to evaluate how these groups (laypersons, technical professionals, and legal experts) make judgments based on their interpretation of legal texts. The survey consists of a legal knowledge test and questions regarding the relationship between similar legal definitions and requirements, and was administered to individuals with varying degrees of legal and IT training and experience. The results show that legal experts achieve stronger majorities (e.g. 80% agreement vs. 60% agreement on a single interpretation) slightly more often than laypersons and technical professionals, and exhibit only slightly greater within-

group agreement overall. Further, we discovered that technical professionals and legal experts achieve consistently higher rates of between-group agreement than laypersons and legal experts.

The remainder of this paper is organized as follows: in Section II, we review related work; in Section III, we present our model of legal decision making; in Section IV we introduce our statistics for measuring consensus; in Section V, we describe our survey design, including question selection; in Section VI we describe our summative results; in Section VII we discuss our findings, including differences in justifications provided by members of each group; in Section VIII we address threats to validity and how they were mitigated throughout the study; and in Section IX we conclude and discuss future work.

II. RELATED WORK

The significance of regulatory compliance in requirements engineering and system development has attracted increasing attention in academic research and industry [30, 12]. In this section, we review techniques and methods in requirements engineering to analyze and understand legal texts, including those that inspired this work; the role of statutory interpretation in determining legal coverage and standards; and studies regarding legal expertise.

Law has become an increasingly important consideration in the requirements engineering community. Research in this area has addressed a variety of issues, including extraction of legal requirements [4, 5], ambiguity detection and resolution [3, 33, 16], and compliance determination [23, 37]. However, these texts do not focus on the interpretive differences made by the analysts due to differences in background. Our approach in this work is driven by earlier studies pursued by the authors in requirements watermarking and requirements coverage modeling, which involve determination of high legal standards in the presence of laws from multiple jurisdictions and differences in legal coverage in the presence of regulatory and environmental change, respectively [18, 19]. In the case of requirements watermarking, differences in interpretation could affect what requirements and coverage conditions contribute to the high standard of care, potentially resulting in a different, though not necessarily incorrect, notions as to what as to what the high standard is. With respect to coverage modeling, differences in interpretation could affect how and when an organization responds to changes in its circumstances.

Legal experts consider a number of factors in addition to the meaning of the legal text when they advise a client about the legal standards they must adhere to. These factors generally relate to the text itself and the current legal landscape, including the novelty and public visibility of relevant laws, the strictness of their enforcement, and the severity of penalties and sanctions imposed. With regards to the text, the legal expert may take into account its legislative history, the intent surrounding its creation, and "lessons of common sense and policy" [13]. The expert will further develop her understanding of the text by taking into account interpretations made by governing bodies or legal authorities and how other members

of the client's industry have chosen to interpret and respond to the text, among other factors. However, while many factors may be considered, the history, legislative intent, and landscape surrounding the text have little weight without first imparting some meaning to the text itself. Some legal experts, such as current Supreme Court Justice Antony Scalia, argue in favor of textualism: that statutory interpretation should begin and end with the apparent meaning of the statutory language [34]. The results of this study should be interpreted within this scope, elegantly encapsulated by Oliver Wendell Holmes' statement, "we do not inquire what the legislature meant; we ask only what the statute means." [22].

Individuals with little or no legal training frequently encounter legal texts, particularly contracts [32]. While it is commonly known that legal documents are difficult for laypersons to understand [21], to the best of our knowledge little research has been conducted investigating what laypersons believe the texts to mean with regards to legal coverage and compliance, much less comparing interpretations made by laypersons to those made by legal experts. Related is a study conducted by Christensen [7], who compared how legal experts and novices read judicial opinions. However, Christensen's focus was not on interpretation, but on reading processes (e.g. was the text read linearly or non-linearly), and she did not present her criteria for determining legal expertise. Of course, expertise can be difficult to define [11], despite being a topic of interest for over a century [35, 36]. Common measurements used to approximate expertise include the presence of experience, certifications, or social acclimation; the ability to make finely discriminable and consistent judgments, and performance on knowledge tests [36]. For example, legal expertise may be informed by possession of Juris Doctor or other certification (e.g. professional paralegal, certified legal assistant, etc.); bar certification in one or more jurisdictions; graded performance on commonly administered authoritative legal examinations, such as the Multistate Bar Examination [29]; years spent practicing law; or through self-attestation. We measured many of these variables in our study, which are detailed in Section V.

Thus, while several requirements engineering researchers aim to equip analysts with tools to demonstrate compliance, little is still known about the differences between experts and novices in how they interpret the law. In the next section, we describe our assumptions that underpin our study.

III. MODELING LEGAL COVERAGE AND REQUIREMENT RELATIONSHIPS

We inform our survey design by adopting a simple model for comparing legal coverage and requirements, which is used to evaluate the judgments obtained through our survey. Consider a small, specialized healthcare clinic based in New York that uses a custom mobile application for employees to record and view vitals (pulse, blood pressure, etc.) with their own mobile devices. They consider opening another branch in California and need to know if California's laws are similar to New York's laws with regards to the types of entities, data, and circumstances covered. Second, they need to know if any new

requirements imposed by California law exceed or fall short of those imposed by New York law. After obtaining relevant laws, they discover that they appear to apply to different types of data, as shown by the requirements in Figure 1.

| NEW YORK | CALIFORNIA |
|---|---|
| [the organization] must maintain a medical record for every person evaluated or treated as an inpatient, ambulatory patient, emergency patient or outpatient | [the organization] must maintain a medical record on all patients admitted or accepted for treatment |

Fig. 1. Medical record requirements from N.Y. Comp. Codes R. & Regs. tit. 10, § 405.10 and California Code Regs. tit. 22 § 70751

Believing the two requirements have some degree of similarity (i.e. they are not disjoint), with regards to data types alone, they consider the following possible coverage relationships, conceived as Venn diagrams:

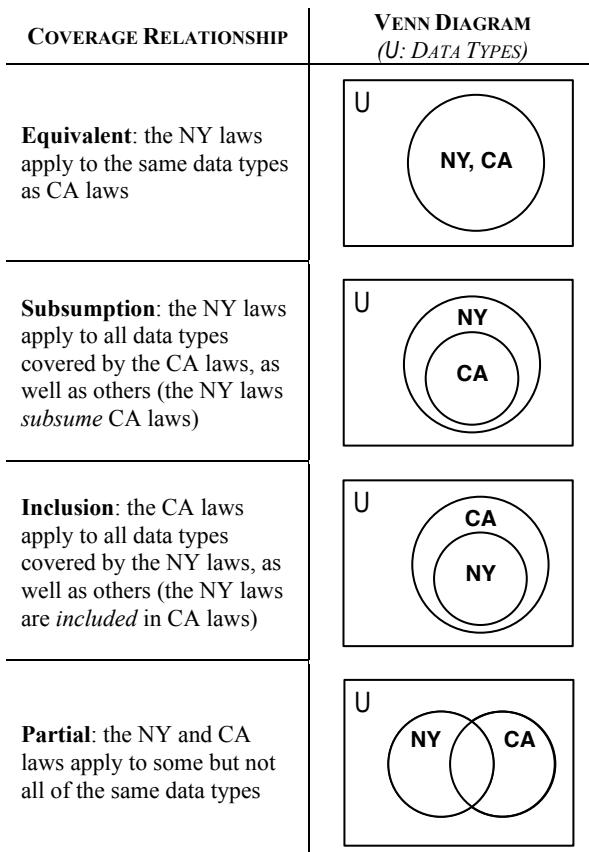


Fig. 2. Possible Coverage Relationships between Data Types Described by NY and CA Law

Ideally, the coverage relationship determined by the clinic would be in agreement with that made by a legal expert. Unlike the clinic, the legal expert's decision would be informed by training and experience enabling her to provide more informed and consistent analyses and decisions than those without [36]. Disagreements about coverage could lead to needless system development constraints and resource expenditures on

unnecessary compliance measures if the clinic believed itself to be covered when the expert believes it is not. Similarly, the organization could unknowingly neglect its obligations and partake in prohibited actions if the clinic believed itself not to be covered when the expert believes it is. In technical terms, such disagreements could include applying excessive data security protections, such as two-factor authentication, or sharing potentially sensitive information with unauthorized 3rd parties. Requirements can also be modeled in this fashion, with the standard of care imposed by one requirement being equal to that imposed by another, subsuming another, etc. Similarly detrimental outcomes could also occur if the clinic made incorrect assessments with regards to the relationship between the requirements as well: for example, if the clinic believed that a New York requirement matched or exceeded the standard of a similar California requirement when the expert believes the opposite, the clinic would fail to take the additional measures necessary to achieve compliance with the California standard.

IV. MEASURING BETWEEN-GROUP AGREEMENT

The extent to which individuals of a particular group agree to item categorization can be measured using an inter-rater reliability statistic, which is frequently used in medicine and psychology. Today, preference is given to statistics that account for the proportion of agreement corrected for chance agreement. Two of the most widely used statistics include Cohen's κ (Kappa) [9] and Fleiss' intra-class coefficients [15]: the former is used to calculate agreement between two raters for a fixed number of items across multiple categories; the latter is for more than two raters. In our survey, we are interested in comparing agreement within and between groups of raters. While methods exist to measure between-group agreement across ordinal data [27, 14], methods are only recently emerging to measure between-group agreement on categorical data [38]. In this study, we determine between-group agreement using a consensus-based Cohen's κ and Van Belle's Nominal Agreement Index (NAI).

The commonly used [38] consensus method involves determining the group consensus (i.e. the mode, or most common categorization) for each item, treating the group's consensus as an individual rater's categorization, and calculating agreement using a common agreement statistic – in this case, Cohen's κ . Though intuitive, by reducing each group to its consensus categorizations, the statistic is imperfect in that it treats the consensus categorization as a binary variable (agree or disagree) rather than a proportion, and does not reflect the *degree* of consensus each group achieves.

Recently developed in 2009, Van Belle's Nominal Agreement Index (NAI) is intended to be a "natural extension" of Cohen's κ for two groups of raters [38] and in the case that each group contains a single rater, reduces to Cohen's κ . The NAI is similar to Cohen's κ by taking into account the relative observed agreement between raters as well as the probability of chance agreement, but with the following two modifications for groups: (a) unlike Cohen's κ , wherein agreement is binary – the two raters agree or do not agree – the NAI's agreement

between groups is based on the *proportion* of each group that assigned a specific category to a specific item, which also allows comparisons between groups of different sizes; and (b) unlike Cohen's κ , wherein the best possible agreement for two raters is 1, the NAI defines the best possible agreement for each item as the category with the highest proportion of raters in agreement. For example, consider Table I that presents two five-member groups 1 and 2 who categorized 10 items into three categories A, B, or C. Each rater's categorizations and group consensus for each item are shown in Table I. As can be seen in the bottom two rows, the group's consensuses are never in agreement, resulting in a simple percent agreement of 0% and Cohen's κ of -.23, indicating the groups agree less than can be expected by chance alone. While the groups' consensus for each item is never the same, their responses are fairly similar, differing by a single rater in all cases. This similarity is accounted for by Van Belle's NAI, which is .78 for the data.

TABLE I. Artificial data illustrating differences in Consensus Cohen's Kappa and Van Belle's NAI

| RATER | ITEM | | | | | | | | | |
|----------------------|------|---|---|---|---|---|---|---|---|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| GROUP 1 | 1 | B | C | A | B | A | C | B | C | B |
| | 2 | B | C | A | B | A | C | B | C | B |
| | 3 | A | B | A | C | C | C | A | C | C |
| | 4 | A | B | B | C | C | A | A | B | C |
| | 5 | A | B | B | C | C | A | A | B | C |
| GROUP 2 | 6 | A | B | B | C | C | A | B | B | C |
| | 7 | A | B | B | C | C | A | B | B | C |
| | 8 | B | C | B | B | A | A | B | B | B |
| | 9 | B | C | A | B | A | C | A | C | B |
| | 10 | B | C | A | B | A | C | A | C | B |
| GROUP 1 CONSENSUS | A | B | A | C | C | C | A | C | C | C |
| GROUP 2 CONSENSUS | B | C | B | B | A | A | B | B | B | B |

Although this statistic has received some expert review, it remains to be empirically validated. Thus, we present our results with both the consensus-based Cohen's κ and the NAI.

V. RESEARCH METHOD AND SURVEY DESIGN

We now discuss our study design, including research questions, survey construction, dataset selection process, participant recruitment methods, units of analysis, and analysis procedure. To guide our research, we proposed the following research questions:

- RQ₁:** Do laypersons, technical, and legal experts make internally consistent judgments regarding relationships about legal coverage and requirements?
- RQ₂:** Do laypersons, technical, and legal experts make similar judgments regarding relationships about legal coverage and requirements?
- RQ₃:** How do laypersons and technical experts justify their interpretations of legal coverage and requirement texts as compared to legal experts?

To address our research questions, we developed and administered an online survey to participants of varying legal and technical backgrounds.

The authors considered numerous research instruments prior to selection of the online survey. Although online surveys may restrict the freedom of participants' qualitative responses as well as the researcher's ability to explore these responses further, unlike interviews, we believed these consequences to be outweighed by the advantages of greater participant convenience and affordability. Legal experts' time is both highly limited and expensive, with even junior associates commanding rates upwards of \$400/hour [20]. Unlike interviews, even if conducted remotely, the online survey gave legal experts the ability to participate in the study when and where they chose without any additional intervention on the part of the researchers. Even given these circumstances, the lack of legal expert availability we encountered motivated multiple means of recruitment as described in subsection C.

A. Survey Process

The survey was administered online in two stages to avoid mental fatigue on the part of the participant and to reduce confounds due to priming effects from the knowledge test: in stage one, participants provide basic demographic information (age, gender), information regarding their technical and legal backgrounds (discussed in Section V) and then answer a number of multiple choice questions obtained from publicly available, multi-state bar examinations to provide basic validation regarding their professed legal background. At the end of stage one, participants are given the opportunity to participate in the second stage, consisting of twenty questions pertaining to requirements comparison and definition-based legal coverage, both framed using the model presented in Section III, and preceded by examples of both question types. Each stage was designed to take approximately 30-45 minutes, and all participants were reimbursed \$10 per stage for a total of \$20.

B. Knowledge and Skills Question Selection

The knowledge questions in stage one were obtained from three publicly available, multi-state bar examination tests in the United States. The multi-state bar examination (MBE) is updated each year by the National Conference of Bar Examiners and is required for admission to the bars of all but two United States jurisdictions [29]. Topics covered include Constitutional law, contracts, criminal law and procedure, evidence, real property, and torts. Twenty questions were down-selected from the 600 available using the following process: (1) after consultation with a legal expert, real property and evidence were removed as categories due to domain specificity; and (2) only questions with one acceptable answer were retained, excluding questions with multiple valid answers. These two criteria yielded a total of 407 remaining questions, from which five were randomly selected from each remaining category (constitutional law, contracts, criminal law, torts) for a total of 20 questions. Order of answer responses was randomized for each participant.

The questions for stage two regarding requirements and legal coverage comparison were constructed by the investigators using requirements and definitions (hereafter referred to collectively as terms) selected from laws analyzed in prior work [17, 18, 19] and chosen because they cover multiple jurisdictions, address the increasing modernization of the healthcare industry, and reflect the most recent developments in data protection, as presented in Table II.

TABLE II. Recent Developments in Data Protection Law

| NAME | YEAR(S) | JURISDICTION |
|--|-------------|-----------------|
| Data breach notification laws | 2003 – 2011 | US; state-level |
| Data destruction laws | 2003 – 2011 | US; state-level |
| Medical record retention laws | 1971 – 2011 | US; state-level |
| Health Insurance Portability and Accountability Act (<i>HIPAA</i>) | 1996 | US; national |
| Health Information Technology for Economic and Clinical Health Act (<i>HITECH</i>) | 2009 | US; national |
| Information Technology Rules | 2011 | India; national |

The terms from these prior studies have undergone extensive analysis, including demarcation, modal classification (e.g., as obligations, permissions, or prohibitions), categorical annotation (e.g. data breach criteria, medical record storage, medical record transfer), identification of internal and external cross-references, and alignment of fully and partially equivalent pairs of requirements or definitions [17]. Each requirement and definition pair was assigned a unique identifier and then randomly sampled for use in the survey. After sampling was completed, terms were reviewed to ensure they were understandable when taken out of context; terms requiring additional context were discarded and replaced.

Importantly, this study is not intended to address an individual's ability to decompose and comprehend elaborately structured definitions (e.g. HIPAA's definition for "covered entity") or ornate requirements with overlapping, clause-heavy exceptions: it is intended to address differences in individual interpretations for short terms or phrases. Even taken in isolation, the requirements and definitions used in this study are verbose, dense, and filled with legalese and jargon. To address this issue, the investigators reviewed the selected terms and conservatively omitted or generalized extraneous text in order to focus the participant's attention on the interpretation of specific words or phrases for each comparison. Details and validity concerns regarding this process are addressed in Section VIII. While some might assume there is a "correct" relationship for each pair of terms that can be treated as a golden standard, we did not design our survey with this assumption.

Finally, terms were framed as comparison questions (example in Figure 3, below) using the relationship types described in Section III, with full, plain-language explanations of each relationship type offered at the top and bottom of each page.

| Consider the following legal definitions: | |
|--|---|
| DEFINITION A | DEFINITION B |
| [an entity] that owns, licenses or maintains computerized data that contains personal information | [an entity] that owns, licenses or maintains personal information |
| Which of the following statements do you believe best reflects the relationship between the above definitions? | |
| (a) Equivalent | (b) Subsumption |
| (c) Inclusion | (d) Partial |

Fig 3. Example Question Comparing Two Definitions

C. Participant Recruitment

Due to the difficulty of recruiting legal experts, participants were recruited through a variety of media, including human intelligence tasks (HITs) on Amazon Mechanical Turk, e-mail solicitations of graduate technical departments of major academic institutions, posts to online message boards (e.g. craigslist), and distribution of fliers at the 2014 IAPP Privacy Summit in Washington, D.C.

VI. SUMMATIVE FINDINGS

Responses were collected over an eight-week period beginning in January 2014. Table III presents the demographic, education and training background of the participants, including the number in each category (Count), their average age, gender as a proportion of females to the total count and education level achieved as a proportion of number of participants in the group possessing the degree to the total number of participants in that group (e.g. 33% of laypersons held bachelors degrees). The training is a self-reported measure of the number of years of experience in information technology, software engineering, computer science, or computer engineering (for technical training) or law (for legal training). Of the 53 participants with complete responses, 25 (47%) came from technical backgrounds with an average of 6.6 years of training and 5.0 years of experience. Technical professionals were on average younger and more educated than laypersons. Legal experts had an average of 5.3 years of legal training and 11.3 years of experience, and all possessed Juris Doctors, except for one English resident possessing an LLB (Bachelor of Law), which is often used to practice law outside the United States.

TABLE III. Demographics, Education, and Training/Exp. by Group

| | LAY | TECH | LEGAL | ALL | |
|--------------------------|----------------|------|-------|------|-----|
| Count | 21 | 25 | 7 | 53 | |
| Avg. Age (years) | 39 | 32 | 42 | 36 | |
| Gender (% female) | .52 | .28 | .29 | .38 | |
| EDU. | Associates (%) | .19 | .24 | .14 | .21 |
| | Bachelors (%) | .33 | .80 | 1.00 | .64 |
| | Masters (%) | .00 | .32 | .86 | .26 |
| | Doctorate (%) | .00 | .08 | .00 | .04 |
| Tech. Training (years) | 0.0 | 6.6 | 2.4 | 3.4 | |
| Tech. Experience (years) | 0.0 | 5.0 | 1.4 | 2.5 | |
| Legal Training (years) | 0.0 | 0.3 | 5.3 | 0.9 | |
| Legal Experience (years) | 0.0 | 0.0 | 11.3 | 1.5 | |

* Three participants encountered computer issues during the survey; their survey times have been excluded from the average.

Table IV describes the MBE test results, including mean, max, and minimum scores by group and subject, as well as the average total time members of each group spent on the text. Because each question had only one correct answer, the score is computed as a simple ratio of correct answers to total number of questions. While the legal experts on average outperformed laypersons and technical professionals on the small sample of MBE questions, the top performers in the layperson and technical professional groups achieved scores higher than the lowest-performing legal expert. All groups performed best on questions regarding tort law. There are a number of potential explanations for this result, such as individuals lacking legal training or experience having established some degree of competence through other means, or specializations among legal experts that depart from areas of law relevant for the MBE. We intend to further explore these results as well as alternative means to measure legal expertise through knowledge testing.

TABLE IV. MBE Performance by Subject and Group

| | LAY | TECH | LEGAL | ALL | |
|-------------------------|--------------------|-------|-------|-------|-----|
| Mean | .39 | .39 | .52 | .41 | |
| Max | .55 | .60 | .70 | .70 | |
| Min | .20 | .15 | .30 | .15 | |
| SUBJ. | Tort Law | .52 | .52 | .66 | .54 |
| | Contracts | .38 | .42 | .63 | .43 |
| | Constitutional Law | .33 | .29 | .46 | .33 |
| | Criminal Law | .32 | .35 | .34 | .34 |
| Average Time* (min:sec) | 26:07 | 29:59 | 19:50 | 27:10 | |

Figures 3 and 4 indicate participants' self-ratings regarding technical and legal knowledge and expertise. Despite having more domain experience than technical professionals, no legal expert claimed to be an expert in their field, whereas many technical professionals did¹. As there are accepted but not definitive criteria for expertise, we wished to acknowledge the possibility, albeit remote, that an individual may be an expert without this status being reflected in other measurements, such

¹ We light-heartedly refer to this phenomena as "nerd arrogance".

as training. In the event this occurred, the individual would be contacted to validate their claim and obtain further information regarding their background. This did not occur in our study.

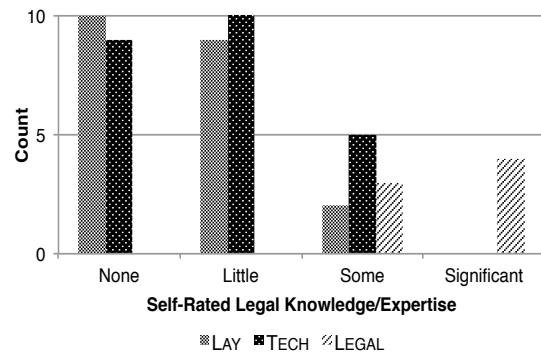


Fig 3. Self-Rated Legal Knowledge/Expertise by Group

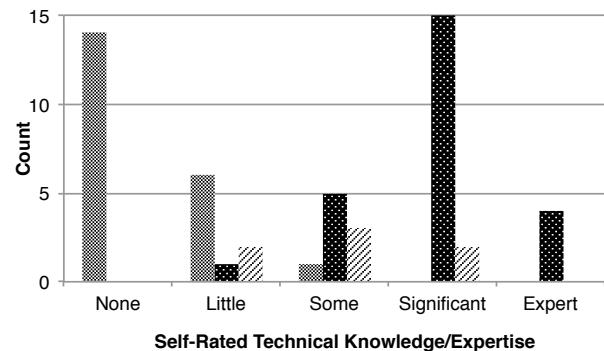


Fig 4. Self-Rated Technical Knowledge/Expertise by Group

Additionally, participants were asked how frequently they read laws and statutory texts, ranging from multiple times a day to never; results are shown in Figure 5. This was done to check for the possibility that an individual may consider herself to have no legal experience but spend a significant amount of time working with such texts, which could also contribute to their ability to function as a legal expert. Laypersons that claimed to read laws and statutory texts "a few times a week" all self-rated themselves as "Little" or "None" on the legal knowledge and expertise scale shown in Figure 3.

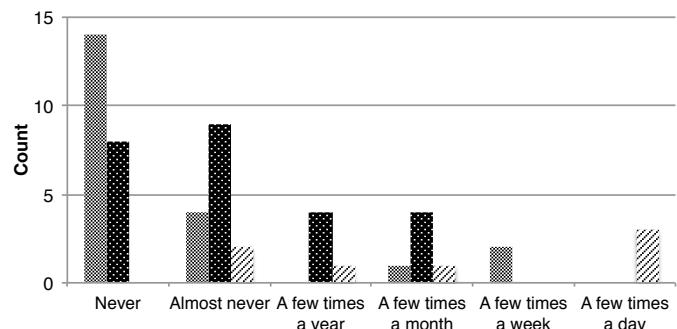


Fig 5. Self-Attested Frequency of Reading Laws and other Statutory Texts by Group

VII. QUANTITATIVE RESULTS

Regarding RQ₁, we first assessed each group's within-group consistency by determining how frequently the group achieved a certain consensus percentile; results are shown in Table V. Each cell in this table indicates the number of comparison questions out of 20 answered by that group that achieved a consensus within the range in the leftmost column: e.g., legal experts achieved a consensus between 60-50% on 6 (30%) of the questions. Legal experts were the only group to achieve perfect consensus, doing so for 3 (15%) questions, and in general achieved higher consensuses more often than the other groups, indicating a greater uniformity in interpretation. The final column is not a total for each row.

TABLE V. Percent of Questions in Consensus Percentile by Group

| CONSENSUS RANGE | COMPARISON QUESTIONS (COUNT; OUT OF 20) | | | |
|-----------------|---|------|-------|-----|
| | LAY | TECH | LEGAL | ALL |
| > 90% | 1 | 1 | 3 | 0 |
| 90 - 80% | 1 | 1 | 0 | 2 |
| 80 - 70% | 2 | 3 | 5 | 3 |
| 70 - 60% | 6 | 4 | 0 | 1 |
| 60 - 50% | 4 | 2 | 6 | 3 |
| 50 - 40% | 4 | 6 | 5 | 8 |
| < 40% | 2 | 3 | 1 | 3 |

In Table VI, we show the inter-rater agreement calculated by Fleiss' κ , which differs from consensus measures used in Table V by factoring in the frequency of all responses for each question rather than just the mode. The possible range of values for Fleiss' κ is from 0 to 1, where 0 means no agreement beyond that expected by chance and 1 means complete agreement. Each row reflects the type of comparison made (between coverage conditions or requirements) with the latter broken down into obligations and prohibitions. All groups achieved fair to poor agreement [1] for each question type, with legal experts achieving greater agreement than laypersons and technical professionals regarding coverage questions ($\kappa = .22, .27$, and $.34$ for laypersons, technical professionals, and legal experts, respectively) but not requirements. In general, groups achieved equal or greater agreement on coverage questions than requirements questions. It should be noted that not all disagreements are qualitatively equivalent, as described in Section VIII.

TABLE VI. Within-Group Agreement (Fleiss' κ) by Comparison Type and Group

| QUESTION TYPE | AGREEMENT (FLEISS' KAPPA) | | | |
|----------------|---------------------------|------|-------|-----|
| | LAY | TECH | LEGAL | ALL |
| Coverage | .22 | .27 | .34 | .23 |
| Requirements | .23 | .20 | .19 | .19 |
| - Obligations | .25 | .19 | .12 | .18 |
| - Prohibitions | .17 | .20 | .28 | .18 |

When addressing the extent of agreement between groups (RQ₂), we first examined how often each group assigned each relationship category across all questions (see Table VII) as a means to describe high-level agreement. The frequencies of each category are similar between groups, with subsumption the greatest and partial the least. Technical professionals assigned the partial category more so than other groups, particularly laypersons (20% compared to 13%). We verified that participants comprehended the directed relationships (e.g. subsumption, inclusion) by checking that their supplemental explanations did not conflict with their given categorical assignment.

TABLE VII. % of Relationship Type by Group

| RELATIONSHIP | LAY | TECH | LEGAL | ALL |
|-----------------|-----|------|-------|-----|
| Equal (%) | .24 | .23 | .24 | .24 |
| Subsumption (%) | .35 | .32 | .35 | .34 |
| Inclusion (%) | .27 | .25 | .23 | .26 |
| Partial (%) | .13 | .20 | .18 | .17 |

While Table VII indicates that the groups assigned each relationship with similar frequency, the consensus-based Cohen's κ and Van Belle's NAI shown in Table VIII both offer a more detailed look at the agreement between groups, as they account for the co-occurrence of these assignments. Question types are the same as those found in Table VI but have been abbreviated for space: COV corresponds to Coverage, REQ_{ALL} to all requirements, REQ_O to Obligations, and REQ_P to Prohibitions. Measurements ranges are identical to Fleiss' κ and interpreted similarly, with a measurement of 0 for no agreement beyond that expected by chance to 1 for complete agreement. Each column in Table VII corresponds to a group (e.g., LA~L compares laypersons to legal, etc.) with the rightmost column comparing laypersons to technical professionals, collectively, and legal experts. Results using a consensus-based Cohen's κ show that for all question types there is greater agreement between technical professionals and legal experts than laypersons and legal experts, with all group pairs showing higher between-group agreement on coverage questions compared to requirements questions. With regards to requirements questions, we discovered that, although there is low agreement between laypersons and legal experts, technical professionals agree with both laypersons and legal experts equally, indicating that they agree with each differently and function as intermediaries. The results using Van Belle's NAI slightly higher on prohibitory requirements. This is because the statistic accounts groups with low within-group agreement for this question type (see Table VI).

TABLE VIII. Between-Group Agreement by Group and Relationship Type using a Consensus-based Cohen's κ and VanBelle's Nominal Agreement Index (NAI)

| CONSENSUS COHEN'S KAPPA | Q. TYPE | GROUP PAIR (LA: LAYPERSONS; T: TECH; L: LEGAL) | | | |
|-------------------------|--------------------|---|-------|--------|------------|
| | | LA ~ L | T ~ L | LA ~ T | (LA,T) ~ L |
| CONSENSUS COHEN'S KAPPA | COV | .42 | .72 | .43 | .57 |
| | REQ _{ALL} | .19 | .46 | .46 | .32 |
| | - REQ _O | .17 | .33 | .38 | .36 |
| | - REQ _P | .27 | .64 | .56 | .27 |
| VANBELLE'S NAI | COV | .44 | .62 | .74 | .55 |
| | REQ _{ALL} | .42 | .56 | .73 | .51 |
| | - REQ _O | .36 | .59 | .83 | .48 |
| | - REQ _P | .71 | .75 | .88 | .77 |

VIII. QUALITATIVE RESPONSES

In order to answer RQ₃, participants were required to provide open-text explanations for half of the comparison questions. Although the explanations given by laypersons and technical professionals were consistently shorter than those provided by legal professionals (averaging 148, 113, and 192 characters, respectively) some responses focus on the same words or phrases in a question (e.g. “paper-based method” in requirement A could mean...) and provided similar justifications. In this section we provide a brief overview of the notable differences in explanations offered by our legal experts as well as those by laypersons and technical professionals.

A. Legal Expert Explanations

Most distinctive of the explanations given by legal experts were those provided when claiming two terms had a partial relationship. Unlike non-legal participants, whose explanations tend to identify specific words that justify the partial relationship, the explanations offered by legal experts not only identify such words but also provide hypothetical examples to illustrate. Legal experts' explanations frequently recognize or hypothesize that words were “terms of art,” such as the various patient classifications referenced in Section III, which we reproduce here:

| REQUIREMENT A | REQUIREMENT B |
|--|---|
| [the organization] must maintain a medical record for every person evaluated or treated as an inpatient, ambulatory patient, emergency patient or outpatient | [the organization] must maintain a medical record on all patients admitted or accepted for treatment |

In this and similar instances, legal experts were reluctant to claim equal, subsumptive, or inclusive relationships without specifying the additional information needed; for example, in reference to the term “*reasonable belief* of unauthorized acquisition of personal information”, one legal expert stated comparison depends on whether an “accusation of a breach that

never actually occurred” would count as a reasonable belief. Some laypersons and technical professionals handle these cases more simply, such as by claiming that requirement B applied to a broader class of patient due to having more cases present.

B. Non-Legal Explanations

Non-legal participants were more likely to interpret different words or phrases as having equivalent meanings, if they believed these words to have the same underlying intent. For example, when considering the two requirements below, which more than half of non-legal participants claimed were equivalent, many participants explicitly stated the two phrases mean “the same thing”, one even stated “I imagine lawyers get a kick out of constructing arguments for why the phrases ‘in the most expedient time possible’ and ‘without unreasonable delay’ might mean different things. I am not one of those people”:

| REQUIREMENT A | REQUIREMENT B |
|---|--|
| [the organization] shall make the disclosure in the most expedient time possible | [the organization] shall make the disclosure without unreasonable delay |

Perhaps most interestingly, some explanations offered by technical participants demonstrated the participant's reliance on formal logic to make comparisons. For example, when comparing definitions that used conjunctions, participants referenced “the lack of associativity of mixed Boolean operators” and set theory in their explanations. One participant went so far as to explain his answer symbolically as follows: “1) $x \text{ or } (y \text{ and } z) = (x \text{ or } y) \text{ and } (x \text{ or } z)$ 2) $(x \text{ or } y) \text{ and } z = (x \text{ and } z) \text{ or } (y \text{ and } z)$ ”.

IX. THREATS TO VALIDITY

We now discuss threats to validity and our mitigations.

A. Construct Validity

Construct validity reflects whether the measurements actually measure that which they are intended to measure [39]. Requirements and definitions used in comparison questions were obtained using previously validated methods for legal requirements extraction [6], have appeared in prior published work [17, 18, 19], and were reviewed by the authors before final inclusion in the survey as described in Section V. As there is no widely agreed definition for legal or technical expertise, we drew on multiple measurements, including experience, training, certifications, and knowledge testing. Similarly, within- and between-group agreement were measured using multiple statistics, with the consensus-based approach in particular chosen to reflect how small firms absent legal resources may attempt to mitigate this shortcoming by combining interpretations from multiple individuals.

B. Internal Validity

Internal validity is the extent to which a causal relationship exists between two variables or whether the investigator's inferences about the data are valid [39]. In this study, the technical participants recruited averaged approximately 10 years younger than the participants in other groups, which

could serve as an alternate explanation for differences in agreement. As the younger average for technical professionals was due primarily to a much higher representation of individuals in the 21-26 age range than in layperson and legal expert groups, we recalculated all statistics using a subset of participants that excluded this age group. In so doing, we found that all conclusions presented in this work remain the same regardless of the presence of age as a factor.

C. External Validity

External validity is the extent to which findings generalize [39]. In this study, we used data obtained from a single domain of data privacy and security, which is less settled than other legal domains, such as tax law [24]. The data sanitization process mentioned in Section V involves omission and generalization of words or phrases in legal definitions and requirements in order to focus our analysis on fewer, direct comparisons, which may have oversimplified the space of interpretation. In all instances, this causes the resultant terms to become more similar, meaning that unsanitized data may show fewer equal categorizations than exhibited in our dataset. In future work we hope to explore differences in interpretations in other legal domains, as well as into the methods used to clarify and subsequently align unfamiliar terms, such as the use of a medical dictionary when analyzing laws regarding medical records.

Legal experts represent only 13% of our sample, compared to 47% and 40% for technical professionals and laypersons, respectively. We believe this is largely due to the value of their time: recruiting 25 experts for a more representative sample using their average rate of pay of \$400 per hour would cost \$10,000 [20]. This is a major concern for research going forward, and motivates alternative means for gaining access to legal experts or finding adequate experimental proxies, such as paralegals or senior law students. While our study is qualitative in nature, we believe the findings offer deeper insight than what could have been obtained using a different study design.

X. DISCUSSION AND SUMMARY

In this paper, we present the results of an exploratory study investigating the role of legal expertise in the interpretation of legal texts governing data privacy and security. To do so, we administered a survey to laypersons, technical professionals, and legal experts in which they were asked to categorize the relationship between related pairs of legal definitions and legal requirements extracted from domestic and foreign laws regarding data protection, medical record privacy, and data breach notification. We discovered that while legal experts achieved greater degrees of within-group consensus (e.g. 70% agreement or greater) on their categorizations more often than laypersons or technical professionals, their overall agreement was only marginally higher whether categorizing definitions (Fleiss' K: .34, .27, .22) or requirements (Fleiss' K: .28, .20, .17). More significantly, we also found that technical professionals and legal experts show considerably greater between-group agreement than laypersons and legal experts, particularly when categorizing definitions (consensus-based

Cohen's K: .72 v. .42). These results are encouraging, suggesting that there may be value in technical professionals performing basic preliminary analysis of laws that affect their system design and development decisions.

In evaluating the explanations that participants offered for their responses, the authors discovered differences in approaches used by each group that merit further investigation. For example, legal experts often justify differences through case-based reasoning, providing instances of each term as evidence to support similarity and dissimilarity. This is in contrast to technical professionals, one of whom employed Boolean logic-based reasoning, which involves treating terms as abstractions. Should further study discover that the instance-based approach is common among legal experts, technical professionals may aim to achieve greater agreement by adopting this approach. Alternatively, legal experts and law firms working with technical clientele may better justify and communicate their opinions by mapping instances to abstract categories. The presence of mixed approaches supports the need for individuals with inter-disciplinary backgrounds that are capable of using both to serve as intermediaries between technical and legal functions.

The findings of this study provide a number of directions for the research community to pursue, including (i) identification of suitable proxies for legal experts in experimental settings, (ii) deeper investigation into the characteristics of statements and laws that produce substantial disagreement within and between different groups, (iii) assessment of weights and considerations given to external or non-textual matters affecting interpretation (e.g. legislative history, case law), (iv) determination of the collective effects different interpretations have on subsequent system design, and (v) further validation of existing between-group reliability statistics.

ACKNOWLEDGMENT

We would like to thank a number of individuals for their advice and guidance regarding legal matters, survey design, and statistics, including Melanie Teplinsky, Baruch Fischhoff, and Will Frankenstein, among others. We would also like to thank the International Association of Privacy Professionals (IAPP) for allowing us to recruit survey participants through their Global Privacy Summit. This research was supported by the Hewlett-Packard Labs Innovation Research Program (Award #CW267287) and National Science Foundation (NSF) IGERT Award #0903659.

REFERENCES

- [1] D. Altman, *Practical Statistics for Medical Research*, Chapman and Hall, 1991.
- [2] Baker & McKenzie, *Global Privacy Handbook*. 2013.
- [3] D.M. Berry, E. Kamsties, M.M. Krieger. "From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity," Univ. of Waterloo Technical Report, November 2003.
- [4] T.D. Breaux, M.W. Vail, and A.I. Anton, "Towards Regulatory Compliance: Extracting Rights and Obligations to Align

- Requirements with Regulations," Proc. 14th IEEE Int'l Requirements Eng. Conf., pp. 46-55, 2006.
- [5] T.D. Breaux, A.I. Anton, K. Boucher, M. Dorfman, "Legal requirements, compliance and practice: an industry case study in accessibility." IEEE 16th Int'l Req'ts Engr. Conf., pp. 43-52, 2008.
- [6] T.D. Breaux, *Legal Requirements Acquisition for the Specification of Legally Compliant Information Systems*. Ph.D. Thesis, North Carolina State University, 2009.
- [7] L.M. Christensen, "The Paradox of Legal Expertise: A Study of Experts and Novices Reading the Law," B.Y.U. EDUC. & L.J., issue 53, 2008.
- [8] L. Christensen, A. Colciago, F. Etro and G. Rafert, "The Impact of the Data Protection Regulation in the E.U," International Think-tank on Innovation and Competition, 2013.
- [9] J. Cohen, "A coefficient of agreement for nominal scales," Educational and Psychological Measurement , issue 20, 37-46.
- [10] A. Cravens, "A Demographic and Business Model Analysis of Today's App Developer," GigaOm, 2012.
- [11] K. Anders, Ericsson et al., *The Cambridge Handbook of Expertise and Expert Performance*. Cambridge University Press, 2006.
- [12] Ernst & Young. "2006 Global Information Security Survey," November 2006.
- [13] W.N. Eskridge and P.P. Frickey, "Statutory Interpretation as Practical Reasoning," Stanford Law Review, vol. 42, 1990, pp. 321-384.
- [14] P.D. Feigin and M. Alvo, "Intergroup diversity and concordance for ranking data: an approach via metrics for permutations," The Annals of Statistics issue 14, 1986, pp. 691-707.
- [15] J.L. Fleiss, "Measuring nominal scale agreement among many raters," Psychological Bulletin, issue 76, 378-382.
- [16] S. Ghanavati, D. Amyot, and L. Peyton, "Compliance Analysis Based on a Goal-oriented Requirements Language Evaluation Methodology," 17th IEEE International Requirements Engineering Conference, 2009, pp. 133-142.
- [17] D.G. Gordon and T.D. Breaux, "Reconciling multi-jurisdictional legal requirements: a case study in requirements water marking," 20th IEEE International Requirements Engineering Conference, 2012, pp. 91-100.
- [18] D.G. Gordon and T.D. Breaux, "A Cross-domain empirical study and legal evaluation of the requirements water marking method." 20th IEEE Requirements Engineering Journal, 2013, pp. 1-27.
- [19] D.G. Gordon and T.D. Breaux, "Assessing Regulatory Change through Legal Requirements Coverage Modeling," 21st IEEE Int'l Req'ts Engr. Conf., 2013, pp. 145-154.
- [20] S.J. Harper, "The Tyranny of the Billable Hour," The New York Times, 2013.
- [21] J. Hartley, "Legal Ease and 'Legalese'," Psychology, Crime, & Law, volume 6, issue 1, 2000, pp. 1-20.
- [22] O.W. Holmes, "The Theory of Legal Interpretation," Harvard Law Review, volume 12, No. 7, 1899, pp. 417-420.
- [23] S. Ingolfo, A. Siena, and J. Mylopoulos, "Establishing Regulatory Compliance for Software Requirements," 30th IEEE International Requirements Engineering Conference, pp. 47-61, 2011.
- [24] B. Johnston, G. Governatori. "Induction of Defeasible Logic Theories in the Legal Domain," Proc. of the 9th Int'l Conf. on AI and Law, pp. 204-213, June 2003.
- [25] L. Kaufman, P.J. Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis," Wiley Publishing, 1990.
- [26] S. Kerrigan, K.H. Law. "Logic-Based Regulation Compliance-Assistance," Proc. of the 9th Int'l Conf. on AI and Law, pp. 126-135, June 2003.
- [27] H.C. Kraemer, "Intergroup concordance: definition and estimation," Biometrika, issue 68, 1981, pp. 641-646.
- [28] H.C. Kraemer, S.P. Vyjeyanthi, and A. Noda "Dynamic Ambient Paradigms," Tutorial in Biostatistics vol 1., pp. 85-105, 2004.
- [29] National Conference of Bar Examiners, "Multi-State Bar Exam," Online: <http://www.ncbex.org/about-ncbce-exams/mbe/>, 2014.
- [30] P.N. Otto, A.I. Anton, "Addressing legal requirements in requirements engineering." 15th IEEE Int'l Req'ts Engr. Conf., pp. 5-14, 2007.
- [31] Ponemon Institute LLC, "Third Annual Benchmark Study on Patient Privacy and Security," 2012.
- [32] J. Redish, "How to Draft more Understandable Legal Documents," from Drafting Documents in Plain Language. 1979.
- [33] A. Rifaut, S. Ghanavati, "Measurement-oriented comparison of multiple regulations with GRL," IEEE 5th Workshop on Requirements Engineering and Law (RELAWS), pp. 7-16, 2012.
- [34] A. Scalia, K-Mart Corp. v. Cartier, Inc., 108 S. Ct. 1811, 1831-34, 1988.
- [35] J. Shanteau, "Decision making by experts: The GNAHM effect," Decision Science and Technology: Reflections on the Contributions of Ward Edwards, 1999, pp. 105-130.
- [36] J. Shanteau, D.J. Weiss, R.P. Thomas, J.C. Pounds, "Performance-based Assessment of Expertise: How to Decide if Someone is an Expert or not," European Journal of Operations Research, 136, 2002. 253-263.
- [37] A. Siena, I. Jureta, S. Ingolfo, A. Susi, A. Perini, J. Mylopoulos. "Capturing Variability of Law with Nomós 2," 31st Int'l Conf. Conc. Mod., pp. 383-396, 2012.
- [38] S. Vanbelle. "Agreement between raters and groups of raters." Doctoral Dissertation, Universite de Liege. 2009.
- [39] R. K. Yin, Case Study Research: Design and Methods, 4th edition, Sage Publications, 2009.

Evaluating the Business Value of Information Technology

Case Study on Game Management System

Harri Töhönen, Marjo Kauppinen

Aalto University, PO Box 9210, Espoo, Finland
{Harri.tohonen, Marjo.kauppinen}@aalto.fi

Tomi Männistö

University of Helsinki, P.O. Box 68, Helsinki, Finland
tomi.mannisto@cs.helsinki.fi

Abstract—Evaluating the multidimensional and dynamic nature of IT business value is a continuous challenge. This paper examines how system dynamics can be used in evaluating IT business value in a company level. We approach IT business value as a web of impacts, where benefits and sacrifices are ultimately evaluated against company earnings logic. This study is based on an action research and covers a pilot project within two co-operating companies. System dynamics was utilised to construct a value creation model for an existing Gaming Management System. This value creation modelling covered two dimensions: 1) structural evaluation of IT impacts with cause-and-effect models, 2) dynamic evaluation and simulation of value realisation over time. As a result, value creation modelling was able to provide a visual overview of how IT impacts were linked to business value through value paths, and how much and when value was realised. Value creation modelling enabled prototyping of value realisation that can provide value based insights for development activities like requirements elicitation and analysis. The examined approach proved its potential for providing a common language for technology and business parties, thus improving IT business alignment.

Index Terms—IT business value, evaluation, system dynamics

I. INTRODUCTION

Various drivers exist for companies to evaluate the business value of their IT [1]. IT business value guides IT investment decisions, focuses and prioritises development requirements [2], and realistic insight on IT current and potential value is the cornerstone for continuous improvements of IT business alignment. In this paper, we approach the IT business value as an aggregation of benefits and costs. The benefits and costs are the effects of IT impacts both on the process and usage level as well as on the company final performance measurement level. With IT we cover the combination of infrastructure technology, software technology and applications.

IT business value evaluation is challenging and the topic has been on both research and practitioner agendas for more than two decades [3]. Multiple reasons drive the challenges of evaluating IT impacts and estimating the value of those impacts. For example, the impacts of IT can be indirect and the business value realisation is a sum of interrelated factors on different levels of company activities and stakeholders [4, 5]. From the value evaluation point of view, IT benefits and costs are separated in time and space. The above mentioned

challenges of IT value evaluation form the background and the research problem of this study. Our research goal is to explore how *system dynamics* (SD) methods and modelling can be used to better understand the construction and realisation of IT business value in a company context. Illustrated by a case study, we explore *how system dynamics can be used in evaluating IT business value*.

According to the systems thinking approach [6], we treat IT as a part of company level business system and IT business value is evaluated against business system purposes. Business system purpose is approached through the company business model and its focal element, earnings logic. System dynamics (SD) modelling is used to understand the elements and their interconnections for value creation, as well as the dynamics of value realisation over time.

The structure of this paper is the following. Section II (*Related work*) investigates the challenges of IT evaluation from the value realisation and business value viewpoints. The system dynamics (SD) approach is introduced with aspects promoting its suitability for our current problem domain. Section III (*Research design*) presents the research empirical case setup and the way we applied action research methods. The results of this research are formulated through three phases. The first phase, *Problem diagnosis* (Section IV) elaborates the IT business value structure and dynamics as well as investigates the theoretical feasibility of SD for matching the evaluation challenges. The second phase, *Action intervention* (Section V) focuses on the pilot case and how SD was utilised in understanding the value creation mechanisms of the pilot company Game Management System. The third phase, *Reflective learning* (Section VI) summarises the applied framework of matching business system, SD modelling methods and IT value creation as a modelling domain. Reflective learning covers also some important experiences confronted during the value creation modelling iterations. In Section VII (*Discussion*), the results are positioned within the IT/IS research field and implications for both practice and research are presented. Further research is proposed to achieve better understanding of how well and in what kind of circumstances SD is feasible to evaluate the IT business value. Section VIII (*Conclusions*) concludes the contributions of this study.

II. RELATED WORK

A. IT Business Value and Business Model as a Goal Setting

IT business value is defined as the contribution of IT to the company performance [1,3]. In general, the business value subsumes the company goal attainments, required efforts or sacrifices and the economic worth [7]. Thus, when studying IT impacts on business value, both benefits and costs (intangible and tangible) have to be evaluated against the degree of the met business goals, and ultimately that goal attainment is judged by its economic worth for the company.

This study investigates IT at an individual IT application level where the directness or indirectness of the impacts of IT on the business value depends on the purpose of IT. The purpose of IT can focus on transactional, informational, strategic or transformational usage [8], or the position of IT utilisation resides on product or service design, production or delivery phases [9].

We utilise the goal driven principles for the IT business value evaluation and explore the business goals from the business model of a company. According to Teece [10], a business model "*reflects management's hypothesis about what customers want, how they want it, and how the enterprise can organize to best meet those needs, get paid for doing so, and make a profit*". The company business model should reveal the concrete ends that are supposed to be the final targets for IT impacts. Nenonen & Storbacka [11] synthesise business model literature to include following ingredients:

- customer value creation i.e. how the company creates value for the customer,
- earning logics i.e. how the company yields a profit from its operations,
- value network i.e. external relationships for value creation,
- resources and capabilities,
- strategic decisions about competitive positioning and target markets or customers.

From the point of view of economic worth, we consider the company earnings logic as a reference point for reflecting and linking IT business value. Depending on the purpose of IT under evaluation, it is possible to add other elements from the business model as a reference point for IT impact evaluation, although the highest level evaluation target would be on costs, revenue and profits level (i.e. earnings logic).

B. Challenges and Approaches Related to IT Evaluation

Various motives exist for understanding the business value of IT. IT is an asset that requires an investment decision, returns of IT investments are targets for monitoring and improvement planning [12], and IT expected value focuses implementation decisions and prioritises development requirements [2]. The IT impacts on benefits, costs and business value have been studied for decades (see, e.g. [3,12,13]). Although IT productivity paradox can be seen as solved and, based on empirical research, IT undoubtedly has positive overall effects [14, 15], the evaluation of IT value remains a challenge for both researchers and practitioners [14].

A few difficulties in evaluating IT as an investment differentiate it from evaluating traditional tangible assets or human capital [16]. At first, IT is not utilised in a vacuum but IT utilisation is a socio-technological phenomena [12]. The returns of IT are, in most cases, dependent on multiple other factors (complementary factors) at different levels of organisation [3,5,17]. Additionally, the spectrum of IT usage targets is wide, ranging from, for example, transaction efficiency to strategic differentiation issues [8,9,18].

As a summary, the following aspects related to IT impacts and evaluating these impacts are repeated in the literature:

- **Delays and dynamics:** IT benefits and costs are realised over varying and extended period of time [19,20]. Value realisation is dynamic and single snapshot measures do not necessarily provide a realistic view about the business value [17].
- **Complementarity:** IT value realisation is affected by multiple factors enhancing or hindering the impacts [3,5,16], outcomes depend on how IT is used [4,15,22] and how organisational changes related to IT usage are implemented and managed [1,4,17,23].
- **Intangibility:** both benefits and costs can be complicated to identify and quantify, especially when those are related to organisational capabilities or strategic goals [20,23]. Intangibility is, in many cases, related to the broad scope and indirectness of IT impacts [23,24].
- **Indirectness and scope:** the links between IT immediate impacts and observed benefits/costs can form multilevel cause and effect chains [4,5].
- **Measures and units of analysis:** conventional input-output accounting measures and methods fail to cover the complexity of IT impacts [14,25]. Variables being measured are affected by external factors and it is challenging to isolate the direct contributions of IT [20].

Most of the solutions for evaluating the IT business value build on recognising the above mentioned aspects to a certain degree [e.g., 4,21]. Conceptual level solutions, for example, aim for balanced measures combining both tangible and intangible or perceived measures [21,14]. IT usage purpose, e.g. transactional, informational or strategic, can also affect the chosen measures [8,26,27]. In addition to focusing only on technology implementation issues, the existence of many complementary factors affecting IT success (e.g. processes, employee skills) persuade also considering organisational change management issues [4,22]. Many classifications of IT benefit [e.g., 18,28] or cost sources [28] can act as checklists for IT evaluation purposes.

DeLone & McLean Information System success model [29] or IS-measurement impact model [25] are examples of general purpose models that recognise multilevel and chained nature of IT impacts. Examples of practical level evaluation frameworks include balanced IS scorecard [30], Project performance scorecard [32], Benefits Dependency Framework [4] or Systems analysis, accounting and strategy framework [21].

Giaglis et al. [32] suggest that the evaluation of IT benefits is an incremental learning process. This evaluation should begin by first modelling direct and quantifiable benefits, after which it is easier to go deeper with more aggregated and complex benefits, thus gradually making intangible benefits more quantifiable. Chan [17] also sees IT evaluation as a company and target IT specific evolution where the evaluation system itself is developing together with the IT system. Chan suggests that, in addition to asking “*What value do IT investments provide?*”, a related set of questions should also be considered: “*Why, where, when, how, and to whom do these investments provide value?*”. Chan’s set of questions promote a holistic approach for IT value creation, thus motivating our study towards systems thinking and system dynamics.

C. System Dynamics and IT Evaluation

System dynamics is based on systems thinking. According to Senge [33, p.68], systems thinking is “... *a discipline for seeing wholes... a framework for seeing interrelationships rather than things, for seeing patterns of change rather than snapshots...*”. The system is, as defined by Meadows [34, p.188]: “*A set of elements or parts that is coherently organized and interconnected in a pattern or structure that produces a characteristic set of behaviours, often classified as its 'function' or 'purpose'*”. The purpose of the system is related to the system boundaries, which are dependent on the observer’s viewpoints. In addition to system overall purpose, it is also vital to identify the usage purpose for the model [6, p89]. System models are always reflections and abstractions of a real world, and the usage purpose of the model defines the appropriate scope and the level of details in the model.

System dynamics (SD) can be seen as one methodology in the systems thinking approach. SD is used to model system structures and the dynamic behaviour over time caused by the interrelationships within the structures. System modelling is supported by two kinds of models: 1) cause and effect causal loop and feedback diagrams (CLD) [35,36,6], and 2) stock-and-flow diagrams (SFD) for presenting the speed of system internal changes, delays and accumulations of issues [6]. The structural analysis with causal loop diagrams is called *qualitative SD* and dynamic analysis with stock-and-flow diagrams is *quantitative SD* [37-39].

Qualitative SD aims to capture both tangible and hidden assumptions of the factors behind the system under investigation. Qualitative SD can be used alone as a problem structuring and group learning method [e.g., 40], but quantitative SD is able to provide insights into the system dynamic behaviour [35]. Quantitative SD formalises the elements of qualitative SD as variables and mathematical equations. Quantitative SD models can be used to simulate the system behaviour: to test different system configurations, to find leverage points in the system structure, or to perform sensitivity analysis for system variables [6]. Qualitative and quantitative SD is supported by software tools such as Vensim or Powersim.

The idea of using SD as a method for evaluating IT and IT impacts is not new. Wolstenholme [41] encourages “holistic modelling approach” for information system benefits

assessment and uses SD as a framework for studying complex and dynamic interactions, as well as exploring the effects of strategic changes within information system. Clark et al. [42] use SD to test the linkage of information attributes (accuracy, timeliness, relevance and reliability) to economic performance measures (cost, profitability and efficiency). Mutschler et al. [43,44] introduce SD modelling based EcoPOST cost analysis framework for process-aware information systems. In addition to static cost factors, EcoPOST also covers dynamic cost factors (e.g. re-design of business processes) and dynamic impact factors (e.g. process and domain knowledge or end user fears).

Georgantzis & Katsamakas [45] have reviewed the literature that focuses on the combination of information systems and SD. These authors categorise the literature based on the level of examined interactions: how information systems impact individuals, groups, organisations and finally markets. When we reflected our study against Georgantzis & Katsamakas categorisation, our aim was to recognise IT impacts both on individual and group levels while developing an organisational level (i.e. company level) understanding on value creation.

Thus far we have presented a motivation for combining IT evaluation as a problem domain and SD as a tool for approaching these evaluation problems. Although this combination is somewhat familiar on a conceptual level in recent literature, an apparent gap seems to be present in empirical research regarding utilisation and feasibility of SD for IT business value evaluation. Value as a multidimensional concept is also recognised as a central factor for software and IT development and decision making within Value-based Software engineering [2]. However, based on authors’ own industrial experiences and research collaborations, the usage of practically feasible methods for the IT value evaluation during requirements engineering analysis and negotiation phases are scarce. The above mentioned gaps in research and needs from practitioners motivated our research for exploring the utility of SD for providing knowledge about value creation, equally for investment decisions, development priorities and for a baseline for long term improvement plans.

III. RESEARCH DESIGN

A. Case Description and Unit of Analysis

This paper addresses the means and experiences of applying system dynamics for understanding IT business value. This study is conducted in joint collaboration of researchers and practitioners. Practitioners provide research context via two separate but related companies: 1) *Host Company* – the primary research partner 2) *Pilot Company* – the customer of Host Company and the provider of the target IT system for piloting the IT business value evaluation.

Host Company is strongly motivated with the research subject because its own service development and customer projects are involved with improved knowledge about the IT value. The mission of Host Company is to ensure the success of the IT investments for its customers. Its service spectrum covers the lifecycle of IT from the quality of requirements to

ensuring the quality of IT service operations. Host Company is developing its offering to better match the value of its customers' IT benefits and value creation. As part of value based service development activities, in spring 2013, Host Company initiated a pilot program for selected customers. The goal of the program was to develop and test the IT value creation modelling methods together with the customers and to use new value insights to guide and focus IT monitoring and quality assurance solutions.

Finnish national betting agency was selected as the first Pilot Company. This Pilot Company provides lotteries and sports betting games. As a target IT system for piloting, the Game Management System (GMS) was chosen. GMS is a support system related to the delivery processes of nearly all Pilot Company game offerings, covering both lottery and sports games. GMS impacts are most tangible with sports games where it covers the entire lifecycle of game delivery, starting from the creation of wagers, defining odds, changing, closing and managing the wagers, as well as handling the gaming results and winning data. During its many years of evolution, GMS has substituted several specific applications and considerable amount of manual work processes. Despite its criticality on operative gaming processes, Pilot Company representatives faced challenges in arguing and concretising GMS maintenance and development decisions in terms of tangible business benefits and value. One of the pilot goals was to provide Pilot Company better knowledge about the value of GMS, in order to make more informed decisions about further investments.

The unit of analysis in the study is the utilisation of system dynamics for understanding the business value of the Pilot Company GMS system. This unit of analysis covers pilot preparation activities within Host Company, pilot implementation activities in a shared context of Host and Pilot companies and, finally, the pilot retrospective within Host Company (see Fig. 1).

B. Research Approach and Process

Our research goal was to explore how system dynamics can be used in evaluating IT business value. The research partner (Host Company) had a practical need for value evaluation solutions, and the preliminary research before this study had uncovered the potential of SD for linking the IT impacts and the business value evaluation.

Action research was selected as a research approach to acquire understanding of SD utility in an actual pilot case. Action research aims to solve current practical problems while expanding scientific knowledge [46, p55]. As the action research is conducted in real life settings together with the practitioners, it is ensured that the action part focuses on practically relevant issues. The research part separates the action research from consultancy, meaning that the problems investigated are also of interest to other researchers and the lessons learned from problem solving can be generalised to relevant theories [46, p.62-63]. This study follows an iterative, three phase research process suggested by Avison et al. [47] and applied in the case context as depicted in Fig. 1. The reported phases occurred from March 1, 2013 to December 31,

2013 and the pilot phase in Pilot Company was conducted in four months June 1 – September 30.2013.

Problem Diagnosis Phase validated the research relevancy both from the practical and research points of view. For practical relevancy, SD modelling prototyping and Host Company interviews were used. For research relevancy, literature studies showed the potential of linking IT evaluation challenges and SD, but also highlighted the research gap for approaching IT business value with means of systems thinking.

| Phase | Description | Context & methods |
|---------------------|--|---|
| Problem Diagnosis | Analysing current situation and defining the problem | <ul style="list-style-type: none"> • Interviews • Prototyping • Literature reviews <div style="border: 1px dashed blue; padding: 5px; margin-top: 10px;"> Context: Host Company <ul style="list-style-type: none"> • Workshops • Modelling iterations </div> <ul style="list-style-type: none"> • Interviews • Pilot retrospective • Literature reviews |
| Action Intervention | Planning improvement actions and implementing the planned actions | |
| Reflective Learning | Analysing the effects of the improvement actions and identifying learnings | |

Figure 1. Research activities and methods of the study.

Action Intervention Phase focused on piloting the business value evaluation for GMS system of Pilot Company. This phase included planning and executing the series of three workshops and iterating the value creation modelling together with Host and Pilot Company teams.

Reflective Learning Phase focused on analysing the experiences of the pilot. Literature reviews were continued for reflecting and reporting the SD modelling experiences with the existing research knowledge. Due to space constraints, the reported experiences focus on value creation modelling substance issues, not on the facilitation of modelling process.

All three phases involved researchers and Host Company team (*Host Team*) close co-operation. Action intervention involved also shared efforts from Pilot Company team (*Pilot Team*). *Host Team* included the Project Manager and the Senior Analyst in execution roles and the Service Production Manager and the Customer Account Manager in steering group roles. *Pilot Team* had five members from different GMS stakeholder roles: gaming business owner, gaming operations and management, IT management, gaming business development and GMS R&D. The research team had two persons, one participating actively in all research phases and in all pilot execution tasks, and the other researcher who participated in two (out of three) reflective learning interviews and in the analysis of the pilot retrospective.

C. Data Collection and Analysis

Data collection methods covered interviews (10), workshops (3), working hour statistics, prototyping, meeting memos and offline communications. Problem Diagnosis Phase included seven interviews with Host Company employees who were working in quality assurance, system analyst or project management positions. The interviewed employees had an

average of 19 person years working experience in the IT field. In Reflective Learning Phase, three group interviews were conducted: one with Host Team and two with Pilot Team. Problem diagnosis interviews were conducted by one researcher and two out of three reflective learning interviews had two interviewing researchers.

Three workshops together with Host and Pilot Teams during Action Intervention Phase were used for gathering input data for the iterative value creation modelling and for evaluating both the modelling outputs and the facilitation process of modelling. Each workshop had formal feedback questions at the end of these sessions. All the interviews and workshops were recorded, transcribed and analysed with ATLAS.ti analysis software. Data analysis continued in writing a retrospective report for Host Company and in the writing process of this paper. All interview and workshop analyses were reviewed together with the researchers.

IV. PROBLEM DIAGNOSIS

Host Company had a goal for grounding their solutions – with reasoning, focusing and pricing - increasingly on customers' IT capabilities on value creation. In order to enable this, Host Company wanted to develop skills and robust practices for linking IT impacts to different organisational levels and finally to business value. The solution development of Host Company had identified the following practical requirements for the IT evaluation approach:

- Methodical – when grounded on the existing methods and practices, it would be easier to develop and maintain the required skills and build prescriptive evaluation guidelines.
- Communication support – visual aids for structuring complicated situations would facilitate learning, knowledge sharing and commitment within different parties.
- Tool support – availability of existing tools for applying the evaluation methods would enhance the solution delivery capabilities.

Based on the initial value evaluation prototypes, systems thinking principles and SD were seen as potential candidates for the value evaluation methods base. When considering from the methodical and tool perspectives, SD is supported by modelling and analysis tools such as Vensim or Powersim, text books are available [e.g., 6] and active user communities exist (e.g. www.systemdynamics.org). Empirical experiences about SD applicability for group learning and knowledge sharing is also available [e.g., 40, 48].

Host Company employees were interviewed to understand the current state of: how customer systems were evaluated, what kind of practical challenges were present and how IT success evaluation and valuing were implemented. As a result, interviewees saw the complexity of IT as a real challenge. Another observation was that the IT project success was very seldom evaluated by the realised value of the IT, but most often on the level of the IT project budget, schedule as well as the technical and use level requirements. Despite of this current status, interviewees saw that their customers were interested in

longer time horizon with IT success evaluation, but the value based evaluation was perceived as very challenging. Finally, the interviews revealed the lack of company level common methods for value evaluation.

The complexity of IT impacts and evaluation of the business value of these impacts was seen as a challenge both by Host Company and in literature [e.g., 5, 12, 27]. One of the goals for the IT business value evaluation was to link benefits and costs from different organisational levels together. Referred to that goal, the complexity problem was simplified as: *IT impacts on benefits and costs are often separated by time and space*. 'Separated by time' means that the IT impacts are observed with varying delays and these delays can be different for benefits and costs. Delays are caused, for example, by multiple organisational levels, cause-and-effect chains and cumulation between the IT first hand impact and the actual measuring point of an observer who is interested in certain benefits and/or costs. These chains of effects and different organisational observation locations represent structural complexity, i.e. 'separated by space' perspective.

The above mentioned simplification regarding the complex nature of IT business value evaluation was used as a basis for considering the feasibility of SD modelling. Together qualitative and quantitative SD would cover both angles of 'separation by time and space' complex nature. By focusing on the business system structure, i.e. elements and their relations, qualitative SD would help in answering **how** IT impacts on value creation. Quantitative SD would be the tool for understanding and simulating the dynamic value realisation over time, thus answering **how much** value and **when**.

Both evaluation prototypes and literature identified challenges with SD usage. The causal loop diagramming provides only qualitative representation of the feedback structure of the system and to obtain more rigorous insights into the system behaviour, quantitative SD with simulation is needed [35]. However, quantitative SD models can be very demanding in terms of modelling skills, time, resources, and availability of data [36]. Availability and uncertainties of data are often related with soft variables, which can be difficult to formulate as equations and for which numerical data is not easily available [38]. Coyle [38] raises the concern that uncertainties of simulation may lead to seriously misleading results. Although, according to Homer & Oliva [37], even in those cases with high uncertainties, formal simulation can provide valuable insights by indicating what kind of information would be required in order to make firm conclusions possible.

Another challenge is related to system and modelling boundaries. Defining the model boundaries and deciding what is treated as endogenous variables (i.e. explanations for phenomena which arise within the system) and what as exogenous variables (i.e. arising from outside), is one of the first activities in the system modelling [6, p.95]. Failures in expanding the boundaries of our mental models lead to too narrow system model boundaries. Too narrow model boundaries, either regarding the time horizon or cause and effect interferences, may restrict us to recover the most

influential structures causing important behavioural patterns [49]. However, broadening the system scope increases the required work effort and the required reference data for modelling. Thus, finding the right balance for the system model boundaries seems to be a critical task. According to Sterman [6, p.87], setting the system boundaries and modelling details is an iterative learning process including modelling, testing and analysing the insights from the models.

Based on the insights gathered during Problem Diagnosis Phase, SD was seen as a feasible method in complex situations to highlight dynamic and counter-intuitive effects with non-linear delays and when continuous modelling approach would be preferred over discrete event based modelling [41,50]. Due to its iterative nature, SD utilisation cost-benefit ratio was assumed to improve in settings where modelling and its benefits could be linked to IT lifecycle thinking. The above mentioned criteria were used in Action Intervention Phase for selecting the pilot case and the target IT system (GMS).

V. ACTION INTERVENTION

Action Intervention Phase focused on testing the SD utility for evaluating the business value of Pilot Company Game Management System (GMS). SD modelling was used to build a value creation model. The value creation model covered both structural and dynamic views to value: 1) **how** IT impacts value creation, i.e. the system elements and interactions within the elements, and 2) **how much** and **when** value is realised over time. In practice, this value creation model was a combination of qualitative causal loop diagrams (CLD) and quantitative stock-and-flow diagrams (SFD). The goals for the value creation model were 1) for Pilot Company to illustrate and concretise GMS role and importance for the gaming business, 2) for Host Company to provide experiences on how value creation modelling would guide measuring and monitoring of the IT value creation.

The piloting was scheduled with three workshops and modelling iterations between these workshops. The topics for the workshops were: 1) Getting the overall picture of the business system with GMS as a part of that system, understanding goals for the pilot and setting initial system boundaries, 2) System structure (elements and relations) validation and focus for further modelling, 3) System dynamic behaviour validation and metrics for value monitoring. The following sections represent how SD modelling was linked with Pilot Company business system.

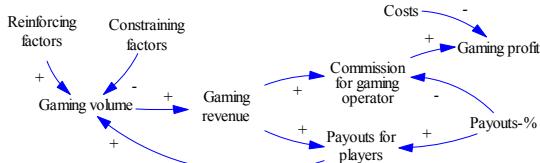


Figure 2. Gaming business earnings logic model. Plus and minus signs indicate the link polarity: '+' for a change to the same direction and '−' for a change to the opposite direction.

The modelling target was a business system that was initially divided into three layers: 1) earnings logic of Pilot Company gaming business, 2) central elements for

implementing or supporting the earnings logic (e.g. processes, functions, phenomena), 3) GMS IT services supporting or enabling the central elements. The earnings logic was used as a starting point for modelling, thus ensuring that the IT impacts would be linked to actual business level goals and valuing measures. Figure 2 represents an earnings logic of Pilot Company gaming business. The gaming volume increases the gaming revenue which in turn is shared between the gaming operator and the players. The gaming profit is the remaining share after the costs are subtracted from the share of the gaming operator.

The following phase identified those elements which either implement or support the execution of the earnings logic. Gaming management and delivery process were used as a source for finding the elements which can link GMS impacts to earnings logic. For example, the most tangible impact of GMS to gaming revenue was realised through the creation and management of wagers. GMS decreased the manual work within the gaming process, which in turn allowed larger gaming target volume compared to the fictional situation where GMS would not be in place. Examples of more indirect impacts of GMS for gaming revenue were through the improved quality of wagers or through decreased human errors in the gaming process.

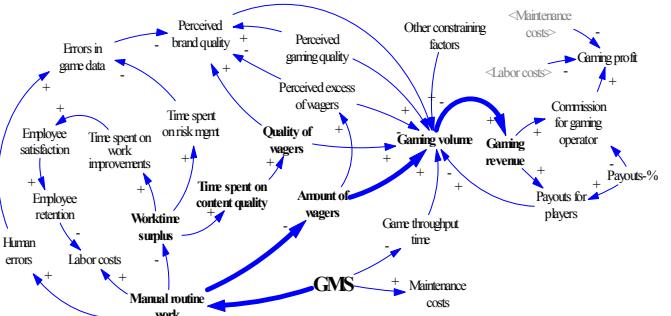


Figure 3. Snapshot of the structural model for linking GMS impacts with business system. Example value path is bolded.

Figure 3 represents a simplified version of the GMS value creation model on a cause-and-effect structural level. The model also includes the visualisation of a *value path*. Value path was used as a concept for representing the impact chain of elements in a value creation model, from IT to a specific end point valuing element. Such an endpoint is used to make the benefits tangible, for the benefits to be further comparable and linkable to the costs of creating that benefit. The value path example in Fig. 3 is related to creating extra sales by the improved efficiency in the creation of wagers. The impact chain for the value path goes through the amount of manual workload, amount of wagers, gaming volume and finally to the gaming revenue. Another value path example covers GMS impacts on the quality of the wagers. In order to have a sustainable gaming revenue, Pilot Team reasoned that the gaming volume should be based on both a proper amount of available wagers and a proper quality of wagers (e.g. topicality and attractiveness of odds for wagers). The balancing of volume and quality was an example of systemic interaction that was recognised from the structural level value creation model.

The previous example of structural value creation model with value paths answered how GMS impacts business value. In order to understand how much and when this value was realised, modelling was continued on a dynamic level. In dynamic modelling, the elements of structural model were presented as variables and enhanced with stock-and-flow characteristics. Furthermore, the relationships between the elements were represented as mathematical equations.

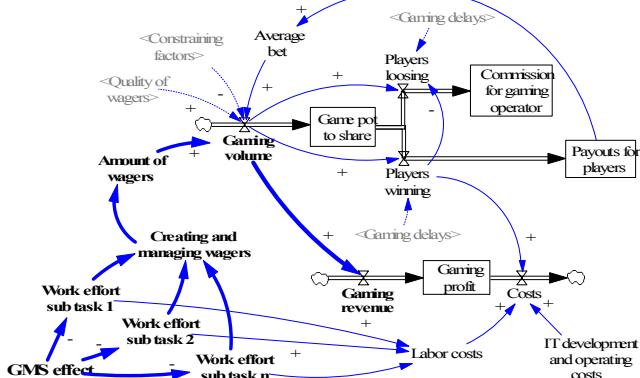


Figure 4. Snapshot of the dynamic model for representing GMS impacts on earnings logic level. Rectangles (stocks) are dynamic elements the values of which are determined by the accumulated changes over time. Arrows with values represent inflows and outflows affecting the values of the stocks.

Figure 4 illustrates how GMS impacts on the gaming process workload and the amount of wagers are linked to the gaming revenue and the gaming profit. The dynamic model snapshot covers the value path illustrated in Fig. 3. The workload data needed in dynamic modelling was based on Pilot Team's estimations for two scenarios: how much work effort in hours is required for running the gaming process with GMS and how much work would be required to achieve the same output without help from GMS. Based on these two scenarios, the dynamic model was used to simulate GMS impacts on gaming revenue. The same model was also used to simulate the costs of compensatory work caused by GMS downtimes. Simulated cost and value estimations provided concrete ideas of possible magnitude of monetary GMS value.

VI. REFLECTIVE LEARNING

Reflective Learning Phase focused on analysing the experiences from the GMS piloting phase and reflecting the learnings against literature. Both Host Team and Pilot Team were interviewed in order to obtain feedback on the pilot workflow issues and the suitability of systemic approach. Workflow issues covered workshop facilitation, communication and resourcing aspects. Systemic approach elaboration covered issues such as intuitiveness of used terminology and relevancy of outputs. In the following analysis we focus on the systemic approach and SD usage experiences as well as piloting outputs.

Piloting outputs. From the outputs point of view, Host Team considered piloting a valuable proof-of-concept for the value creation modelling. SD was a feasible method in concretising IT business value and identifying measures for

monitoring value realisation. Pilot Team saw the structural value creation model with value paths as the most important output. The structural model was able to provide a value based overview of the role of GMS within the business system. The overview was appreciated in communication between the technical and business parties.

The framework. Figure 5 summarises how SD was utilised for the IT business value evaluation. SD provided the basic methods for reflecting the reality of business system into the structural and dynamic models of value creation. The analysis of the business system was initiated from the business level goals, which were discovered from the business model and especially from the earnings logic. The business system analysis was continued by inspecting processes, usage and IT services for linking the IT impacts with value creation. The value creation model was implemented on two interconnected levels of value structures and dynamic value realisation. The qualitative SD with causal loop diagrams facilitated structural modelling and the quantitative SD with stock-and-flow diagrams enabled dynamic modelling and simulations.

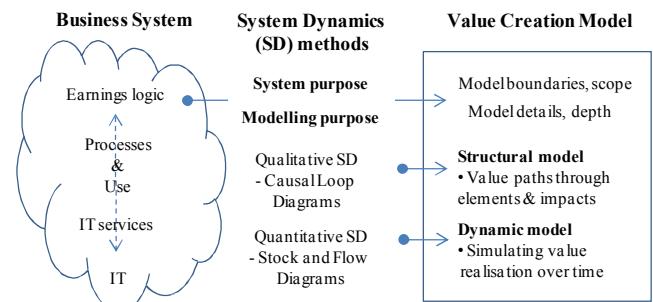


Figure 5. The framework for utilising SD for IT business value evaluation.

Importance of purposes. Two kinds of purposes influenced the scoping and focusing of modelling: 1) the purpose of the system, and 2) the purpose of the modelling itself. The purpose of the system was identified from the earnings logic, and that ensured the IT impact elaboration against the business value. In general, setting the system boundaries defines the highest order value evaluation level that can be reached by modelling the system under evaluation. By calling the system under evaluation a business system, the expectations for the purpose of the system are set to the correct level for the IT business value evaluation.

The purpose of modelling was closely related to the question *“how good is our model, should we continue further or is this enough?”*. The purpose of modelling can be process oriented or output oriented or both. The process oriented purpose promotes knowledge sharing and group learning about the system. The output oriented purpose focuses on the usage of the models in supporting decision making. In the pilot, modelling had both process and output oriented purposes, but the output oriented purpose was not explicit in the beginning. The level of explicitness of the final usage goals of the models affected the level of difficulty when deciding the proper level of details and accuracy in modelling. The purpose of modelling also affected the balance of effort used for structural and dynamic modelling.

Interplay of modelling levels. Based on Host Team and researcher experiences, cost-effective balance between structural and dynamic modelling was an important issue. Using causal loop diagramming for identifying system elements and defining the system structure was an intuitive opening activity. Enhancing structural causal loop diagrams with the dynamic stock-and-flow diagramming supported the understanding of cumulations and delays. When advancing the dynamic modelling further, the usage of data and mathematical equations in stock-and-flow model simulations allowed the testing of different system configurations and the observance of the leverage points. The dynamic modelling also provided feedback to the structural modelling level by indicating fuzziness in cause-and-effect relations or suggesting the division of too abstract elements into more atomic elements. However, the benefits of the dynamic modelling required considerable extra effort both in technical modelling (e.g. semi-programming and debugging Vensim-models) and collection and analysis of the required input data for modelling.

Iterative modelling. The appropriate scope and depth of dynamic modelling was very open in the beginning of the pilot, as the output oriented purpose of the modelling was not set explicitly. During the modelling iterations, simulation gradually focused into one value path; GMS impacts to workload and to gaming revenue. Pilot Team interviews in Reflective Learning Phase revealed the consequences of the partly fuzzy modelling purpose. Some members of Pilot Team were very satisfied with the modelling outputs as they saw the structural cause-and-effect diagrams as the most important and usable result. The other members of Pilot Team were satisfied with the structural model as an overview and communication tool, but they wished that the models were more detailed in order to provide more game type specific insights (e.g. different sport betting games). However, Pilot Team members felt confident for the potential of dynamic modelling and they were ready to invest more work in the future for acquiring better data for more detailed modelling. As the current level of the dynamic model covered only one value path, further modelling iterations would be necessary for understanding the leverage points and the desired balance of, for example, the volume and quality of wagers. The structural modelling of the interactions of these crossing value paths provides only hints about the leverage points and balancing aspects, but the dynamic modelling with simulation would be required for more concrete insights.

As the snapshot of the dynamic model demonstrated (see Fig. 4), GMS impacts were derived to high level business goals. The other possible direction for deepening the model would have been towards specific functionalities and characteristics of GMS. That level of analysis would provide value based information for focusing GMS further development and prioritising the technical requirements. Experiences regarding the gradually growing understanding of feasible modelling scope and details were in line with the notions of iterative and evolutionary nature of system modelling [e.g. 6, p.87].

Importance of visualisation. Pilot Team recognised the importance of visual aspects of the models for guiding and focusing communication and group learning. In the beginning, structural modelling with causal loop diagramming acted also as a conceptual model of important terms and concepts. Because of this, structural diagrams were also called *conceptual* cause and effect models. Pilot Team saw the structural level (see Fig. 3) value creation model with visualised value paths as a potential counterpart for traditional enterprise architecture models. Pilot Team considered the value path concept intuitive which helped in quick understanding of value creation structural model.

VII. DISCUSSION

The research question of this study is *how system dynamics can be used in evaluating IT business value*. The answer was based on seeing the IT impacts on business performance as a combination of structural and dynamic aspects. The structural aspects were examined by asking “How IT impacts value creation, what kind of elements and interconnections are involved?”. The dynamic aspects were approached by asking “how much value is realised and when?”. SD matched the IT business value evaluation by utilising qualitative modelling methods for structural aspects and quantitative modelling with simulation methods for dynamic aspects. In order to ensure that the IT impacts were evaluated against business value, IT was seen as a part of business system where the system purpose was defined by the earnings logic of the company. The IT business value evaluation was conducted by identifying IT impact chains – value paths – from IT towards goals in the earnings logic level. Qualitative SD modelling was used to identify the structure of value paths and quantitative SD for modelling and simulating value realisation over time within value paths. Together structural and dynamic models formed a value creation model.

Positioning and limitations. SD is both a way of thinking and a concrete method, but not a source of content for value evaluation. SD is neutral for system elements and theories or causal logics forming the interrelationships between the elements, and SD is equally neutral whether the elements of the system model are reflections of hard/tangible or soft/intangible measures of the real world. Based on the above positioning, SD is complementing other research and practice based contributions that provide content level insights into the IT usage types [e.g. 26,9] or different IT benefit and sacrifice categories [e.g. 18,24,44]. This study also has a content dimension by proposing company earnings logic as a starting and anchoring point for identifying the business level content for value creation modelling. However, earnings logic is only one possible business level grounding point and most probably other business model areas (such as resources or co-creation network management) can be more natural starting points for other types of IT systems.

Although this report omits SD working process and facilitation aspects, we see that the efficiency of communication and data collection methods as well as adjustable workshop facilitation activities are critical for

successful IT evaluation exercise. Value realisation is dependent on IT usage and the usage is always related to specific company context. The efficiency of identifying and analysing context specific factors in different company settings directly affects the cost-benefit ratio of the value creation modelling.

Based on a single case study, our experiences are limited regarding the balance of qualitative and quantitative modelling. We assume that the explicitness of modelling purpose affects the proper balance and the required modelling iterations. Sterman [6, p.90] emphasises the importance of a reference mode in the very beginning of modelling efforts. The reference mode is a set of graphs or other descriptive data showing the development of a target problem over time. The reference mode is one tangible method for explicitly defining the modelling purpose. Our experiences indicate that several iterations are required until the value creation structures are visible enough for a definition of the reference mode. It is reasonable to assume that the higher level IT (e.g. strategic IT) requires more modelling iterations than the lower level IT (e.g. transactional IT [8]).

Managerial implications. The IT evaluation approach presented in this paper is based on a cross-section of organisational levels from technology to business, thus strengthening the alignment between these two viewpoints. From business performance and investment management points of view, SD based IT evaluation is not replacing the traditional financial measures such as ROI (Return On Investment) or NPV (Net Present Value) analysis. Instead, the presented approach identifies and links the data elements and measures for making the traditional financial measures more representative of the real and dynamic situations. By doing this, SD utilisation helps in opening the 'black box' of IT and enriches the IT business alignment.

From requirements engineering and IT development points of view, the presented approach provides concrete value based insights for focusing and prioritising requirements elicitation and analysis. The value creation model, especially the structural model, provides the development team 'a big picture' where IT can be seen as a part of the business system. Value paths highlight the critical chains that link IT concretely to business value. These critical value paths can be used as one of the drivers when requirements are elicited and prioritised.

Systemic approach encourages the development projects to consider also complementary and transitional factors [17] for value realisation, not merely direct functional and technology oriented requirements. These requirements are no longer solely IT requirements but they are business *system requirements* with responsibilities for value realisation shared within the system. In that case, the entire time horizon of IT lifecycle is considered and the transition from development projects to operations and usage phase is better prepared for the continuous value management.

Prototyping is another application for SD modelling on the development level. Prototyping is a common method for validating requirements and obtaining early feedback for IT development. SD modelling and simulation can be seen as a

consequences oriented alternative or complementary method for functionally oriented prototyping.

Research implications. This study argues for holistic IT value evaluation and proposes concrete methods for approaching the IT business value. Utilisation of SD requires integration to organisational knowledge that is, in many cases, diffused within varied groups of stakeholders. The processes and methods for facilitating knowledge and data assimilation into value creation models frame an important topic for further research. This topic can be approached from multiple levels, e.g. 1) how SD is integrated into IT management practices for better business alignment or 2) how SD is integrated into IT development and long term planning practices.

This study argumented the coupling of IT business value evaluation and SD utilisation. However, our literature studies revealed surprisingly few empirical studies about this combination. This observation raises further questions:

- Why coupling of IT business value evaluation and SD seems to be a rare topic in literature? More thorough systematic literature study should be conducted.
- Is SD embedded in the existing practices? When companies perform IT evaluation, how do they implement it? How do the existing practices match the challenges of the IT business value evaluation?

The above mentioned research suggestions form the basis for our further case studies, which have been initiated at the beginning of 2014. Enhancing the dynamic cost and impact factor patterns of EcoPOST framework [43,44] with dynamic value creation factors, present another interesting venue for further research.

VIII. CONCLUSIONS

This case study explored IT business value evaluation and utilised system dynamics (SD) for studying IT impacts on different organisational levels. The nature of IT impacts on business value were compiled into 1) structural dimension: **how** IT impacts value creation, and 2) dynamic dimension: **how much** and **when** value is realised. SD modelling was used to identify the value paths from IT impacts to business value and for simulating the value realisation over time. SD modelling visualised a map – value creation model – of IT impacts piercing through IT services, processes and usage towards the execution of company earnings logic. The value creation model offered a common language for technology and business parties. Value creation model was seen as a potential tool for promoting value management over the lifecycle of IT: from the value based requirements prioritisation to identification and measuring of enablers and inhibters for value realisation.

REFERENCES

- [1] P. Tallon, K. Kraemer, and V. Gurbaxani, "Executives' perceptions of the business value of information technology," *Journal of Management Information Systems*, vol. 16, no. 4, 2000, pp. 145-173.
- [2] A. Wohlin and A. Aurum, "Criteria for selecting software requirements to create product value: An industrial empirical study," In *Value-Based Software Engineering*, Biffl, S. Aurum, A.Boehm, B., Erdoganmus, H., Grünbacher, P., (eds.) Springer, Berlin,Germany, 2006, pp. 179–200.

- [3] N. Melville, K. Kraemer, and V. Gurbaxani, "Information technology and organizational performance: An integrative model of IT business value," *MIS Q.*, vol. 28, no. 2, 2004, pp. 283-322.
- [4] J. Peppard, J. Ward, and E. Daniel, "Managing the realization of business benefits from IT investments," *MIS Quarterly Executive*, vol. 6, no. 1, 2007, pp. 1-11.
- [5] S. Lee, "Modeling the business value of information technology," *Information & Management*, vol. 39, no. 3, 2001, pp. 191-210.
- [6] J. Sterman, "Business dynamics: Systems thinking and modeling for a complex world," McGraw-Hill, Boston, 2000.
- [7] R. Kauffman and C. Kriebel, "Modeling and measuring the business value of information technology," *NYU Working Paper*, 1988.
- [8] S. Gregor, M. Martin, W. Fernandez, S. Stern, and M. Vitale, "The transformational dimension in the realization of business value from information technology," *The Journal of Strategic Information Systems*, vol. 15, no. 3, 2006, pp. 249-270.
- [9] M. E. Thatcher and D. E. Pingry, "Modeling the IT value paradox," *Commun. ACM*, vol. 50, no. 8, 2007, pp. 41-45.
- [10] D. Teece, "Business models, business strategy and innovation," *Long Range Plann.*, vol. 43, no. 2-3, 2010, pp. 172-194.
- [11] S. Nenonen and K. Storbacka, "Business model design: conceptualizing networked value co-creation," *Int. Journal of Quality and Service Sciences*, vol. 2, no. 1, 2010, pp. 43-59.
- [12] R. Stockdale and C. Standing, "An interpretive approach to evaluating information systems: A content, context, process framework," *European Journal of Operational Research*, vol. 173, no. 3, 2006, pp. 1090-1102.
- [13] S. Mithas, A. Tafti, I. Bardhan, and J. Goh, "Information technology and firm profitability: Mechanisms and empirical evidence," *MIS Quarterly*, vol. 36, no. 1, 2012, pp. 205-224.
- [14] M. Davern and C. Wilkin, "Towards an integrated view of IT value measurement," *International Journal of Accounting Information Systems*, vol. 11, no. 1, 2010, pp. 42-60.
- [15] S. Devaraj and R. Kohli, "Performance Impacts of information technology: Is actual usage the missing link?," *Management Science*, vol. 49, no. 3, 2003, pp. 273-289.
- [16] J. Dedrick, V. Gurbaxani, and K. Kraemer, "Information technology and economic performance," *ACM Computing Surveys*, vol. 35, no. 1, 2003, pp. 1-28.
- [17] Y. E. Chan, "IT value: The great divide between qualitative and quantitative and individual and organizational measures," *J. Manage. Inf. Syst.*, vol. 16, no. 4, 2000, pp. 225-261.
- [18] P. Simmons, "Quality outcomes: Determining business value," *Software, IEEE*, vol. 13, no. 1, 1996, pp. 25-32.
- [19] S. Devaraj and R. Kohli, "Information technology payoff in the health-care industry," *Journal of Management Information Systems*, vol. 16, no. 4, 2000, pp. 41-67.
- [20] A. Ragowsky, N. Ahituv, and S. Neumann, "Identifying the value and importance of an information system application," *Information & Management*, vol. 31, no. 2, 1996, pp. 89-102.
- [21] A. Bajaj, W. Bradley, and K. Cravens, "SAAS: Integrating systems analysis with accounting and strategy for ex ante evaluation of IS investments," *Journal of Information Systems*, vol. 22, no. 1, 2008, pp. 97-124.
- [22] C. Tiernan and J. Peppard, "Information technology: Of value or a vulture?," *European Management Journal*, vol. 22, no. 6, 2004, pp. 609-623.
- [23] S. Kamungo, S. Duda, and Y. Srinivas, "A structured model for evaluating information systems effectiveness," *Systems Research And Behavioral Science*, vol. 16, no. 6, 1999, pp. 495-518.
- [24] Z. Irani, A. Ghoneim, and P. Love, "Evaluating cost taxonomies for information systems management," *European Journal of Operational Research*, no. 173, 2006, pp. 1103-1122.
- [25] G. Gable, D. Sedera, and T. Chan, "Re-conceptualizing information system success : The IS-impact measurement model," *Journal of the Association for Information Systems*, vol. 9, no. 7, 2008, pp. 377-408.
- [26] R. Urwiler and M. Frolick, "The IT value hierarchy: Using Maslow's hierarchy of needs as a metaphor for gauging the maturity level of information technology use within competitive organizations," *Information Systems Management*, vol. 25, no. 1, 2008, pp. 83-88.
- [27] G. Marthandan and C. Tang, "Information technology evaluation: issues and challenges," *Journal of Systems and Information Technology*, vol. 12, no. 1, 1997, pp. 37-55.
- [28] S. D. Ryan and D. A. Harrison, "Considering social subsystem costs and benefits in information technology investment decisions: A view from the field on anticipated payoffs," *Journal of Management Information Systems*, vol. 16, no. 4, 2000, pp. 11-40.
- [29] W. Delone and E. McLean, "The DeLone and McLean model of information systems success: A ten-year update," *Journal of Management Information Systems*, vol. 19, no. 4, 2003, pp. 9-30.
- [30] M. Martinson, R. Davison, and D. Tse, "The balanced scorecard: A foundation for the strategic management of information systems," *Decision Support Systems*, vol. 25, no. 1, 1999, pp. 71-88.
- [31] C. Barclay, "Towards an integrated measurement of IS project performance," *Information Systems Frontiers*, vol. 10, no. 3, 2008, pp. 331-345.
- [32] G. Giaglis, N. Mylonopoulos, and G. Doukidis, "The ISSUE methodology for quantifying benefits from information systems," *MCB University Press*, vol. 12, no. 1/2, 1999, pp. 50-62.
- [33] P. Senge, "The fifth discipline: The art & practice of the learning organization," Doubleday/Currency, 1990.
- [34] D. Meadows, "Thinking in systems: A primer," Earthscan, London, 2008.
- [35] S. Santos, V. Belton, and S. Howick, "Adding value to performance measurement by using system dynamics and multicriteria analysis," *International Journal of Operations & Production Management*, vol. 22, no. 11, 2002, pp. 1246-1272.
- [36] S. Santos, V. Belton, and S. Howick, "Enhanced performance measurement using OR: A case study," *Journal of the Operational Research Society*, vol. 59, no. 6, 2007, pp. 762-775.
- [37] J. Homer and R. Oliva, "Maps and models in system dynamics: a response to Coyle," *System Dynamics Review*, vol. 17, no. 4, 2001, pp. 347-355.
- [38] G. Coyle, "Qualitative and quantitative modelling in system dynamics," *System Dynamics Review*, vol. 16, no. e, 2000, pp. 225-244.
- [39] E. Wolstenholme, "Qualitative vs quantitative modelling: The evolving balance," *J Oper Res Soc*, vol. 50, no. 4, 1999, pp. 422-428.
- [40] J. Vennix, "Group model-building: tackling messy problems," *System Dynamics Review*, vol. 15, 1999, pp. 379-401.
- [41] E. Wolstenholme, A. Gavine, K. Watts, and S. Henderson, "The design of a dynamic methodology for the assessment of computerised information systems," *System Dynamics*, 1990, pp. 1346-1354.
- [42] T. Clark and F. Augustine, "Using system dynamics to measure the value of information in a business firm," *System Dynamics Review*, vol. 8, no. 2, 1992, pp. 149-173.
- [43] B. Mutschler, M. Reichert, and S. Rinderle, "Analyzing the dynamic cost factors of process-aware information systems: A model-based approach," *In: 19th CAiSE*, 2008, pp. 510-524.
- [44] B. Mutschler and M. Reichert, "On modeling and analyzing cost factors in information systems engineering," *In: 19th CAiSE*, 2007, pp. 589-603.
- [45] N. Georgantas and E. Katsamakas, "Information systems research with system dynamics," *Information systems research with system dynamics*, vol. 24, no. 3, 2008, pp. 247-264.
- [46] M. Myers, "Qualitative research in business & management," Sage, Los Angeles, 2009.
- [47] D. Avison, F. Lau, M. Myers, and P. Nielsen, "Action research," *Communications of the ACM*, vol. 42, no. 1, 1999, pp. 94-96.
- [48] F. Stallinger and P. Grünbacher, "System dynamics modelling and simulation of collaborative requirements engineering," *Journal of Systems and Software*, vol. 59, no. 3, 2001, pp. 311-321.
- [49] J. Sterman, "All models are wrong," *System Dynamics Review*, vol. 18, no. 4, 2002, pp. 501-531.
- [50] M. Kellner, R. Madachy, and D. Raffo, "Software process simulation modeling: Why? What? How?," *Journal of Systems and Software*, vol. 46, no. 2-3, 1999, pp. 91-105.

Non-functional Requirements as Qualities, with a Spice of Ontology

Feng-Lin Li, Jennifer Horkoff, John Mylopoulos

University of Trento

Trento, Italy

{fenglin.li@, {horkoff, jm}@disi.}unitn.it

Renata S. S. Guizzardi, Giancarlo Guizzardi

Federal University of Espírito Santo (UFES)

Vitória, Brazil

{rguizzardi, gguizzardi}@inf.ufes.br

Alexander Borgida

Rutgers University

New Brunswick, USA

borgida@cs.rutgers.edu

Lin Liu

Tsinghua University

Beijing, China

linliu@tsinghua.edu.cn

Abstract—We propose a modeling language for non-functional requirements (NFRs) that views NFRs as requirements over qualities, mapping a software-related domain to a quality space. The language is compositional in that it allows (recursively) complex NFRs to be constructed in several ways. Importantly, the language allows the definition of requirements about the *quality of fulfillment* of other requirements, thus capturing, among others, the essence of probabilistic and fuzzy goals as proposed in the literature. We also offer a *methodology* for systematically refining informal NFRs elicited from stakeholders, resulting in unambiguous, de-idealized, and measurable requirements. The proposal is evaluated with a requirements dataset that includes 370 NFRs crossing 15 projects. The results suggest that our framework can adequately handle and clarify NFRs generated in practice.

Index Terms—Non-functional requirements, goal models, software qualities, ontologies

I. INTRODUCTION

Non-functional requirements (NFRs) — such as usability, maintainability, security and performance — have been difficult to deal with since the very beginning of Requirements Engineering (RE) back in the ‘70s. NFRs are known to have a make-or-break status in software development projects, but are difficult to treat formally.

In RE research and practice, NFRs have been treated in one of two ways: (a) they included all requirements that were not functional (hence their name); (b) they were requirements on quality of the system-to-be, such as usability, maintainability and the like. The former approach bundles together very different kinds of requirements and makes it hard to come up with any kind of formal treatment. The latter approach has led to standards like ISO/IEC 25010 [1]. However, these standards do not say much about the exact nature of the qualities nor how to exploit them in dealing with NFRs.

One of the proposals that attempted to deal with NFRs in depth was the NFR framework (NFRF), first proposed in 1992 [2] and extended into a monograph [3]. In this proposal, NFRs were modeled as “softgoals” — goals with no clear-cut criterion for success. The NFRF offered a simple representation that allowed basic reasoning, such as: if I make design decisions A, B and C, how am I doing with respect to softgoal SG? However, goals lacking a clear criterion of satisfaction (i.e., softgoals) turn out to be not always NFRs — most early requirements, as

elicited from stakeholders are also “soft”. For instance, when a stakeholder says “Upon request, the system shall schedule a meeting”, this is also vague and needs to be made more firm: Do we allow requests for any time (e.g., weekends)? Should the system notify participants about the scheduled meeting? Should it account for contingencies (e.g., power outage)? etc. Our conclusion is that softgoals constitute a useful abstraction for *early* requirements, both functional and non-functional, rather than just non-functional ones.

But this conclusion begs the next question: what then are NFRs, how do we model them and how do we use these models in the RE process? We begin with treating them as “*qualities*”, and look to foundational ontologies to tell us precisely what qualities are [4]. Foundational ontologies have been defined in the research area known as Applied Ontology (AO) and they include the most general concepts needed for any domain, such as *object*, *event* and *quality*. Prominent foundational ontologies include DOLCE [5] and UFO [6]. In these ontologies, a quality is defined as an individual (instance) with the power to connect the entity it qualifies (its *subject*) with a *value* in a geometric *quality space*.

NFRs are often specified in idealized and/or vague terms, making it hard to assess their fulfillment. Take, for example, NFRs from the PROMISE dataset [7]:

- *NFR-1: “The product shall return (file) search results in an acceptable time.”*
- *NFR-2: “Administrator shall be able to activate a pre-paid card via the Administration section within 5 sec.”*
- *NFR-3: “The website shall be available for use 24 hours per day, 365 days per year.”*
- *NFR-4: “The interface shall be appealing to callers and supervisors.”*

These NFRs are problematic for a number of reasons:

- *NFR-1* is vague, and therefore not measurable.
- It is unclear whether *NFR-2* is strict or gradable (can be relaxed). E.g., would “5.7 sec.” do? If gradable, what are the constraints on the possible values?
- *NFR-3* is idealized and unsatisfiable as such, given all the contingencies that could render it unfulfilled (e.g., power failures, strikes, government shutdowns, etc.)

- For *NFR-4*, some users may report that the interface is appealing while others do not agree. This is due to the fact that some qualities (e.g., *look, appearance*) can be subjective, resting “*in the eye of the beholder*”.

The aim of this work is to propose a language for modeling NFRs, addressing the challenges listed above. Our proposal makes the following contributions:

- Adopts an ontological interpretation of NFRs based on qualities in foundational ontologies.
- Offers a compositional modeling language for capturing NFRs, where the subjects of involved qualities can be identified using (arbitrarily nested) notation, resembling feature structures in linguistics [8].
- Identifies three (combinative) meta-qualities for talking about the fulfillment of a requirement: universality of a proposed solution, gradability of the fulfillment of the requirement, and agreement among stakeholders that indeed the given requirement is satisfied.
- Proposes a goal-oriented requirements methodology for *refining* ambiguous, (practically or logically) unsatisfiable, or vague NFRs to unambiguous, de-idealized and measurable ones.

The remainder of the paper is structured as follows. Section II introduces the research baseline for this work, section III and IV present a language for capturing NFRs treated as qualities. Section V proposes a goal-oriented methodology for refining NFRs, while section VI evaluates the proposal using a publicly available dataset of requirements. Section VII reviews and correlates related work, and section VIII concludes and offers suggestions for future work.

II. RESEARCH BASELINE

Requirements as goals. Goal-Oriented Requirements Engineering (GORE) is founded on the premise that requirements are goals that stakeholders want to fulfill. Key GORE proposals include seminal work on the KAOS project [9], *i** [10], and Techne [11], as well as the above mentioned NFRF [3].

In GORE, goals can be refined to other goals through AND/OR refinement. In this paper, we distinguish between (i) functional goals that need to be achieved by functions performed by the system or an external actor, and (ii) quality goals that capture qualities of the system. Functional goals are operationalized by tasks/functions, while quality goals are operationalized by quality constraints, as in Techne [11]. For example, “*collect traffic info*” is a functional goal *FG#1* that might be operationalized by tasks “*use fixed sensors*” or “*use mobile phone with GPS*”, while “*collected traffic info in real-time*” is a quality goal related to *FG#1*. As the name suggests, operationalization makes requirements operational [12], either by providing a task that fulfills a functional goal, or by offering a formally specified Boolean constraint that measures whether a quality goal is fulfilled.

Qualities as mappings. Ontologically speaking, a *quality* is defined as a basic perceivable or measurable characteristic that inheres in and existentially depends on its subject [5][6]. The subject can be an object, process, action/task, goal, as well as collectives of objects, processes, and so on. In proposals such

as DOLCE [5] and UFO [6], quality is a particular (i.e., instance), e.g. *cost#1* represents the cost of a specific trip. Each quality has a *quality type* *QT* (e.g., *Cost*), which is associated with a *quality space* *QS* (e.g., *EuroValues*). These approaches also differentiate a quality, e.g. *cost#1*, from its value, e.g., 1000€, which is a point or region in the corresponding *QS* (*EuroValues*).

Our notion of quality space is based on the notion of *Conceptual Space* put forth by Gardenfors [13]. In this theory, quality spaces should be understood literally, given that these structures are endowed with geometrical and topological properties. For instance, associated with the quality type *Cost* we can have a *EuroValues* space, a one-dimensional structure isomorphic to the positive half-line of 2-place decimal numbers; other quality types such as *Color*, *Security* and *Usability* are associated with multi-dimensional spaces, with proper constraints on the constituting dimensions (reflecting the geometry of the space at hand). This theory can be adapted or extended to address a number of relevant conceptual phenomena, from context-dependent, non-monotonic and analogical reasoning [13] to graded membership in vague regions [14].

In this work, we simplify the rich quality theory by treating a quality *Q* (be ontologically correct, *Q* is a quality type) as a mapping (mathematical function) that takes an individual subject *subj* of type *SubjT*, to a quality value (point or region) in *Q*’s codomain (quality space). For example, as a mapping, the quality “*usability*” takes its subject, say a software system “*the E-movie manager*”, to a region “*good*” in its quality space. In RE, qualities are also often applied to entire subject types. E.g. the quality “*processing time*” in *NFR-1* applies to all possible runs of the system. For a fuller account of our ontological treatment of NFRs, interested readers can refer to [14].

III. NON-FUNCTIONAL REQUIREMENTS AS REQUIREMENTS OVER QUALITIES

NFRs as qualities. Adopting a qualities-as-mappings perspective, we model an NFR as a *quality goal* (*QG*) that constrains a quality mapping *Q* to take values in a desired region *QRG* of its quality space for its subject type *SubjT*, and capture a *QG* using the notation in Eq. 1, which is an abbreviation of $\forall x. \text{instanceOf}(x, \text{SubjT}) \rightarrow \text{subregionOf}(Q(x), \text{QRG})$, meaning that for each individual subject *x* of type *SubjT*, the value of *Q(x)* should be a sub-region of (including a point in) *QRG*.

$$Q(\text{SubjT}) : \text{QRG}^1 \quad (1)$$

In addition, we use ‘:=’ to assign names to expressions, for later reference. E.g., *NFR-1* can be modeled as *QG1* in Example 1, below. *Quality constraints* (*QC*) that operationalize *QGs* use the same syntax, but must involve measurable regions. E.g., a corresponding quality constraint for *QG1* is shown in the same example, as *QC1*.

Example 1 (NFR-1).

QG1:= processing time (file search): acceptable.

QC1:= processing time (file search): ≤ 8 sec.

QC1 is-operationalization-of QG1

¹ If *Q* is an aggregate quality like *universality* and *average*, the argument of *Q* will be a set, whose type is a **power-set**, denoted as $\mathcal{P}(\text{SubjT})$. In this case the syntax will be $Q(\mathcal{P}(\text{SubjT})) : \text{QRG}$.

NFRs can be defined over both subject types and individual subjects. In Example 1, the subject “*file search*” is a type, not an individual (in object-oriented terms, a class, not an instance); here we refer to a set of its instantiations, i.e., a set of file searches. The expression of QG1 (QC1) implies a set of QGs (QCs), each of which requires a specific run “*file search #*” to take a processing time value in the acceptable (≤ 8 sec.) region. Hence QG1 (QC1) is interpreted as “*for each file search, its processing time shall be acceptable (≤ 8 sec.)*”.

Consider another NFR: “*The interface shall be intuitive*”. In this case, the subject of the requirement is an individual subject, a singleton: “*understandability* (*{the interface}*): *intuitive*”, where “*understandability*” is a quality, “*intuitive*” is the desired quality region in the corresponding quality space where the ease of understanding is relatively intuitive.

Quality domains and codomains. The concept of quality in DOLCE [5] and UFO [6] refers to a broad category of intrinsic properties of entities that can be projected on a quality space (roughly, the basis of a measurement structure that becomes the codomain of the associated quality mapping [15]). Examples can be found in every domain, including color, shape, length, atomic number, electric charge, etc.

For our purposes, we adopt the quality model proposed by the ISO/IEC 25010 standard [1] as our reference. This standard distinguishes two categories of qualities: *qualities in use* and *product qualities*, with five and eight qualities, respectively. Fig. 1 shows the eight product qualities and their refinements. For example, “*usability*” is refined into “*learnability*”, “*operability*”, “*accessibility*”, etc.

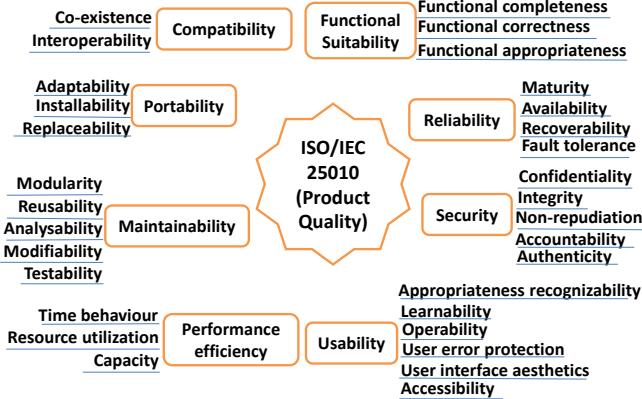


Fig. 1. The eight product qualities in ISO/IEC 25010 (with refinements)

Domains and codomains of qualities are key components in the specification of an NFR. For a specific quality, the set of subject types that it can be applied to constitutes its domain, and the union of all the possible values will form its codomain (quality space).

The domain of a software quality can be any aspect of a software system, including its constituents (code, architecture, requirements, etc.), the software processes that created it, its runtime environment, and the like.

Standards such as ISO/IEC 9126-1 [16] and 25010 [1] are helpful, up to a point, in defining a codomain for qualities. For example, “*availability*” is defined as “*degree to which a system, product or component is operational and accessible when*

required for use”. Hence it will be associated with a codomain that is a scale ranging from 0% to 100%. We show in Table I possible domains and codomains of 10 frequently used qualities in our evaluation experiment (more details can be found in Section VI).

TABLE I. THE DOMAIN & CODOMAIN OF 10 FREUENTLY USED QUALITIES

| Quality | Domain | Codomain |
|----------------------------|-------------------|--|
| Operability | {a system} | {time to operate}; {ease of operating: easy, hard...} |
| Availability | {a system} | {0% ~ 100%} |
| Processing / Response time | {functions/tasks} | {time interval}; {slow, ... fast, ...} |
| Scalability | {a system} | {simultaneous transactions} |
| Learnability | {a system} | {time to learn}; {ease of learning} |
| Frequency | {functions/tasks} | {numbers per time unit} |
| Understandability | {a system} | {ease of understanding} |
| Modifiability | {a system} | {time to modify} |
| Look and feel | {a system} | {degree of preferences} |

The structure of the codomains of some qualities may be complex, and can differ depending on their subjects. For example, according to ISO/IEC 25010, the codomain of *usability* is a six-dimensional space, with each of its sub-qualities being one dimension. Of course, stakeholders may only be concerned with some of these sub-qualities, in which case a *usability* QG should be refined accordingly. For example, if only *learnability*, *operability*, and *accessibility* are of concern, then the codomain of usability becomes three-dimensional.

By differentiating a quality (type) from the quality spaces it can be projected on, we can account for the possibility of having multiple quality spaces (with measurement structures derived from them) for the same quality (type) [15]. Thus such a quality (type) could map an individual subject to different quality values in their respective quality spaces. For example, as shown in Table I, “*learnability*” can map “*a system*” to either “*easy/good*” or “*x minutes of training*” in different quality spaces.

It is important to highlight that our qualities cannot be equated with attributes in the tradition of conceptual modeling [6]. In general, attributes are conventional ascriptions of property values to individuals. Qualities, in contrast, inhere in their bearers, i.e., there is something intrinsic (in the bearers) that makes true a certain property ascription to these bearers. That is, attributes are properties assigned to objects while qualities are properties intrinsic to them [17] (ones we have to design for a system). E.g., “*release date*” and “*serial number*” are merely conventional attributions of certain values to a software system. In contrast, when we state that a system has 50K LOCs or high reliability, there is something *in the system* that makes these statements true.

IV. REPRESENTING COMPLEX NFRS

The preliminary syntax introduced so far, along with a catalogue of qualities allows us to express simple NFRs such as *NFR-1*, but is not sufficient to capture the following aspects of an NFR: (1) a subject restricted by qualifiers, acting as relative clauses, e.g., going from “*activate a pre-paid card within 5*

sec.” to “activate a prepaid card <by Administrator> <via the Administration section> within 5 sec.” (NFR-2); (2) NFRs that are unsatisfiable because of blanket use of universal quantifiers (NFR-3, but also NFR-2); (3) hard constrained NFRs that leave no room for flexibility (e.g., NFR-2); and (4) subjective NFRs whose satisfaction depends on the eye of the beholder (e.g., “appealing look” in NFR-4). To address these issues, we need to enrich our language with new constructs.

A. Qualified Subjects

We extend the basic syntax introduced earlier by allowing its subject type $SubjT$ to be restricted by *qualifiers* that consist of $\langle attribute: filler \rangle$ pairs referring to $SubjT$ (or fillers, when nested)². By using this language, we are able to define particular sets of individual subjects, over which we can talk about concerned qualities. For example, the subject of NFR-2, “activate pre-paid card”, is a software function and can be qualified by the attributes “*actor*” and “*means*”, as in Example 2. It represents the set of activations performed by administrators through the admin section (past or future).

Example 2 (NFR-2).

$activate p\text{-}card' :=$

$activate \text{ pre-paid card } \langle \text{actor: } \text{Administrator} \rangle$

$\langle \text{means: } \text{via the Administration section} \rangle.$

$QG2 := \text{processing time} (activate p\text{-}card'): \text{within 5 sec.}$

B. Qualities of Fulfillment

Many requirements can be represented as logical assertions of the form $\forall x P(x)$, as in “*For every request ($\forall x$) a meeting shall be scheduled ($P(x)$)*” (FR) and “*Every file search ($\forall x$) will be completed within 5 sec ($P(x)$)*” (NFR). Inspired by knowledge representation techniques for uncertainty [18], we propose three meta-qualities on the fulfillment of a requirement: (1) *universality*: the degree to which the set of all x satisfies P ; (2) *gradability*: the degree to which P holds for each x ; (3) *agreement*: the degree to which observers agree P holds for each x . We accordingly define three meta-quality functions, U for universality, G for gradability, A for agreement, that can be applied to requirements, functional or non-functional, to define quality goals.

Universality. The U operator aims at limiting universality for its requirement subjects, in that a requirement need not be fulfilled in all cases, but rather in a percentage thereof. E.g., NFR-3 can be relaxed as “*the website shall be available 99.5% of the time per year*”, expressed as

$theWebsite' := theWebsite$

$\langle \text{at: time units } \langle \text{in-period: a year} \rangle \rangle$

$QG3 := \text{availability} (theWebsite'): 100\% \text{ //the entire unit}$
 $QG3-1 := U (QG3): 99.5\% \text{ //99.5\% of the units in a year}$

U takes as argument a set of requirement subject instances, which is of type **power-set**($SubjT$) (i.e., $\mathcal{P}(SubjT)$), and returns a percentage of the instances for which the requirement is to-be-fulfilled in the linear space $0\% \sim 100\%$. In this case, the subject “*theWebsite*” has N instances representing the system during each unit in a one-year period, and QG3 according-

² Our proposed language currently does not provide a built-in set of attributes, which requires an ontology of software systems and of the application domain.

ly has N QG instances, with each of them requiring the website to be 100% available for its corresponding time unit. Originally, all QG3’s instances are required to hold. It is now relaxed to QG3-1, saying only 99.5% of them need to be satisfied. NFR-2 can be relaxed in a similar way, saying $k\%$ of the activations shall be within 5 seconds.

The U operator regulates/modifies the fulfillment of a requirement, either non-functional or functional, from a universal or statistical perspective. For instance, the requirement “*all users shall be authenticated*” can be represented as a functional goal FG that calls for a function *authenticateUser*. If stakeholders can tolerate some failures for this requirement, say 1%, this can be captured by the universality requirement “ $U (\text{authenticateUser}): 99\%$ ”. During elicitation, it is useful to ask a stakeholder who calls for a universal requirement in the form of “ $\forall x P(x)$ ”, whether he/she really means it for all x : “Could you live with less, and if so, how much less?”. It is also helpful to remind the stakeholder that universal requirements are at the very least harder and more expensive to fulfill, and at worst simply unsatisfiable.

Gradability. The G operator allows for partial satisfaction of a requirement. Specifically, G maps a requirement to its desired degree of fulfillment on a linear scale $0\% \sim 100\%$.

When evaluating the satisfaction of an NFR that specifies either crisp quality regions such as “ $\leq 5 \text{ sec.}$ ”, “ $2 \sim 3 \text{ m}$ ” and “ $100 \sim 200 \text{ €}$ ”, or vague regions like “*fast*”, “*high*” and “*low*”, the measured or perceived quality value may approach but not be exactly located in the desired region. To accommodate degrees of satisfaction, we use G to relax NFRs. For example, NFR-2 (captured as QG2) can be relaxed as QG2-1, requiring the processing time value to be *nearly* within the region (0 sec., 5 sec.], or QG2-2, requiring the processing time to be *90%* in the region. By using G , the membership of a time value in the interval (0 sec., 5 sec.] is made gradable (“*fuzzy*” in the sense of Fuzzy Logic [19]), and we only require a partial membership (e.g., *nearly*, 90%) in that interval for fulfillment. The relaxed membership can be clear or vague; if vague (e.g., *nearly*), it shall eventually be made clear (e.g., 90%) through operationalization. For details on calculating graded membership based on the theory of quality space, we refer to our companion paper [14].

$QG2 := \text{processing time} (activate p\text{-}card'): \text{within 5 sec.}$

$QG2-1 := G (QG2): \text{nearly}$

$QG2-2 := G (QG2): 90\%$

G can also be applied to relax NFRs with vague quality regions besides those with crisp regions like (0 sec., 5 sec.]. For instance, NFR-1 can be relaxed as follows, requiring the processing time value to be *moderately* in the *acceptable* region.

$QG1 := \text{processing time} (\text{file search}): \text{acceptable.}$

$QG1-1 := G (QG1): \text{moderately.}$

The G operator also captures the degree of fulfillment of functional requirements (FRs). A functional requirement, especially one that calls for multiple/batch tasks, can also be only partially fulfilled. For example, if room equipments have not been returned after a meeting, the scheduling requirement can be seen as *almost* but not *totally* fulfilled.

Agreement. Agreement (**A**) is intended to address the subjectivity of qualities. The satisfaction of some NFRs, especially those concerning qualities that depend on human individuals, such as *look*, *attractiveness* and *satisfaction*, is subjective and will vary with the observer who is beholding.

We can make such NFRs objective by operationalization (e.g., “*the interface shall be intuitive*” can be operationalized as “*80% of the new users can operate the system without training*” with the use of **U**), or by using **A** to capture the agreement among observers that a requirement is indeed satisfied. E.g., *NFR-4* in our list can be rephrased as *QG4-1*, requiring 80% of the callers and supervisors to agree that *QG4* holds.

QG4:= look (the interface): appealing

QG4-1:= A (QG4): 80% of the callers and supervisors

A relates to the notion of *precision* widely used in Science and Engineering. Precision for a measuring system is defined as the degree of repeatability, i.e., the extent to which multiple measurements lead to the same result. In our case, the measuring system is an observer and a measurement consists of the observer determining whether a requirement is satisfied or not. In this sense, the domain and codomain of **A** consist of a set of requirements, and ratios of observers from a given pool who agree each requirement is satisfied, respectively.

Composition. In practice, a requirement may be specified using multiple applications of the three operators. For example, to make *NFR-2* practical, we can relax it by using **U**, **G**, or both. As shown below, we first use **G** to relax *QG2* as *nearly* being in the region (0 sec., 5 sec.], then use **U** to relax the set of all executions of “*activate p-card*”, requiring 95% of the activation processes to be *nearly* in that interval.

QG2:= processing time (activate p-card'): within 5 sec.

QG2-1:= G (QG2): nearly

QG2-3:= U (QG2-1): 95%

These three operators can be combined in many different ways: **U** over **G** (firstly apply **G** and then **U**, as in the above example), **A** over **G** (e.g., 80% of the users report the website is *kind of* hard to understand), **G** over **U/A** (e.g., *nearly* 90% of the activation takes 5 sec., *nearly* 80% of the users report the interface is simple), or even **G** over **U/A** over **G** (e.g., *nearly* 90% of the activation takes *nearly* 5 sec.). The full syntax and semantics of proper operator nesting will be part of our future work.

V. A FRAMEWORK FOR GOAL MODELS WITH QUALITIES

We introduce next a goal modeling framework enriched with qualities and a methodology for refining early and informal NFRs to unambiguous, satisfiable and measurable ones. The framework and methodology is evaluated through the case study on the PROMISE requirements dataset [7] in Section VI.

A. Goal Models with Qualities

Our conceptual model for goal models is shown in Fig. 2 and includes the concepts introduced earlier. In general, we represent a requirement as a Goal, which is further specialized into Functional and Quality Goal. Functional goals are operationalized by functions (i.e., tasks), while quality goals are operationalized by quality constraints. Any goal can be opera-

tionalized by a domain assumption. E.g., the functional goal “*Find room for meeting*” may be operationalized by a domain assumption like “*There are enough rooms available for all requested meetings*”. Also, by function constraint, a function can be constrained to situations that must hold before/after/during its execution, analogously to pre/post-conditions and invariants. E.g., “*only managers are allowed to activate users*” is a constraint over the function “*activate users*”. The three kinds of refinements, namely disambiguation, relaxation and focus, will be introduced in detail in the next section.

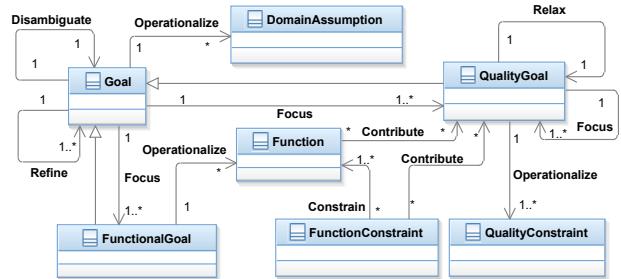


Fig. 2. The conceptual model for the revisited goal modeling framework

B. Building Goal Models during Requirements Analysis

In general, goals elicited from stakeholders are vague, ambiguous, idealized, etc. The aim of requirements analysis is to iteratively refine them into a specification that includes concrete and/or measurable functions, quality constraints and domain assumptions.

Our corresponding methodology is based on iteratively answering the following questions: (1) Is a requirement/goal unambiguous? (2) Is it (practically) satisfiable? (3) How do we make it measurable? In response to these questions, we perform *disambiguation*, *relaxation*, and *focus* refinement, in addition to the usual logical (AND/OR) refinements and operationalization of goal models.

Disambiguation. This is the first phase for capturing and analyzing requirements. On discovering the ambiguity of a given requirement/goal [20], we can keep asking the following questions: (1) What is the subject of the goal? (2) What quality does it refer to (if any)? These questions help identify not only the subject (and the quality) of a goal, but also potential ambiguities. E.g., by focusing on the subject in “*the interface shall have standard menu buttons for navigation*”, we find that there are four possible interpretations: (1) there should be buttons all of which should be standard; (2) there should be buttons some of which should be standard; (3) if there are buttons then all of them should be standard; (4) if there are buttons then at least some should be standard. The customer may choose (1), in which case we should refine the goal to “*the user interface shall have menu buttons, and all of them shall be standard*”. Ambiguity may also arise from word polysemy and multiplicity of structural analyses, interested readers can refer to [20].

Relaxation. After disambiguation, we need to analyze whether a goal is satisfiable or not in practice. As discussed earlier, a requirement may be hard to satisfy because of: (1) the use of universal quantifiers; (2) the specification of categorical quality regions (e.g., “ ≤ 5 sec.”); (3) subjectivity. In

such cases, we use the three operators, **U**, **G**, and **A**, or a combination thereof to relax the requirement to a satisfiable (and acceptable to stakeholders) degree.

Focus. Using focus refinement, we can focus a goal intertwining functionalities with qualities to functional goals and/or quality goals (e.g., “*collect real-time traffic info*” can be stated as a goal, and focused to “*collect traffic info*” and “*timeliness (collected traffic info): real-time*”), or focus quality goals by concentrating on sub-qualities or parts/elements of the system that are of concern. For quality goals, focusing may move along two dimensions: (1) the quality/sub-quality hierarchy, and (2) the hierarchy of their subjects, generalization or aggregation for some subjects, goal hierarchy for goal subjects. The quality hierarchy we use is defined in the quality standard adopted, such as ISO/IEC 25010 [1].

For example, the quality *usability* has sub-qualities *learnability*, *operability*, *accessibility*, etc., according to ISO/IEC 25010. As shown in Fig. 3 (the meeting scheduler case study, MS), the quality goal concerning “*good usability*” (*MS-QG3*) may be focused into *MS-QG4* and *MS-QG5* that require respectively the system to be easy to learn and operate. Similarly, since a meeting scheduler has different functions, we can further apply *learnability* and *operability* to functionalities such as “*set up a meeting*” and “*reserve a conference room*”, obtaining *MS-QG6* and *MS-QG7* respectively.

When applying these two focusing refinements, we should pay attention to the applicability of a quality to a subject. It is not always meaningful to apply a quality to all the parts of its subject, or all the sub-qualities of a quality to its subject. For instance, user friendliness for a system may be focused into user friendliness for its interface, but likely not for its timeslot scheduling component. That is, quality functions are inherently partial w.r.t. a software-related domain.

Note that the two steps, *relaxation* and *focus*, may be interleaved in practice. For instance, the goal $G := \text{“monitor events”}$ may first be focused into $G' := \text{“monitor suspicious events”}$, and then relaxed into $U(G') := 98\%$.

Operationalization. Once quality and subject have been refined to a suitable granularity, we are left with the problem of operationalizing vague quality values, i.e., making them measurable. E.g., if we have a quality goal such as *MS-QG7* in Fig 3, how do we measure whether it is *easy* to learn?

To answer this question, we need to choose one or more quality dimensions (called metrics in ISO/IEC standards) to measure *learnability*. The ISO/IEC 9126-2 standard [21] can help in this respect, e.g., *learning time*, *help frequency*, etc. This means that operationalization may use several dimensions of the quality space of *learnability*.

Let us assume that *learnability* is measured by *learning time*. To get a reference value for *easy*, a typical way is to determine a comparison class, a set of similar meeting scheduler systems, and apply the quality *learning time* to the set of subjects to get a set of typical quality values. We can then get a reference value from these values, say, the average.

When determining typical time values, we need to focus to the exact subject that the quality being considered inheres in, because typicality may be manifested differently as the subject

varies, resulting in values in different regions of a quality space [22]. E.g., typical values for the *easy* learnability region concerning the function *scheduleTimeslot* may be different from those for the function *informParticipants*.

Contribution. In our proposed framework, goals can be focused to functional goals leading to functions, or to quality goals resulting in quality constraints. However, our models do not allow refining a functional goal into a quality goal and vice versa. To address situations where functional elements contribute to the satisfaction of quality goals, we use contribution relationships (*help*, *hurt*, *make*, and *break*) of functions or constraints over functions on relevant quality goals. For instance, the function “*authenticateUsers*” would help the authenticity of a system. Contribution links constitute an important element of tradeoff analysis during RE processes [23] and will be explored in future work.

VI. EVALUATION

The PROMISE (PRedictOr Models in Software Engineering) dataset consists of 625 requirements collected from 15 software development projects [7]. Among them, 255 items are marked as functional requirements (FRs) and the remaining 370 non-functional requirements items are classified into 11 categories, such as *Security*, *Performance* and *Usability*. Classification counts are shown in the second column of Table II.

In this section, we describe a comprehensive case study on the 370 PROMISE NFRs. Our aim is twofold: *a*) to evaluate the need for our framework by examining the nature of NFRs in practice; and *b*) to evaluate the expressiveness of our framework by applying it to the set of NFRs of *meeting scheduler*, one of the fifteen projects in the PROMISE dataset. To evaluate *a*), we observe the occurrence of elements in our conceptual model, and evaluate the implicit use of and need for our proposed meta-qualities. To evaluate *b*), we rewrite the set of NFRs of *meeting scheduler* by using our proposed syntax, applying our methodology as described in Section V.

A. The Necessity of our Framework: PROMISE NFRs

We first classified the 370 NFRs according to our ontological classification of requirements. Our classification includes three basic categories of requirements, “functional requirement (FR)”, “quality requirement (QR)”, and “constraints over function (CF)”. These would be modeled by functional goals, quality goals and function constraints in our conceptual model (see Fig. 2), respectively.

Our classification is shown in Table II. Among the 370 items, we identified 187 QRs, 52 FRs, 50 CFs, 61 requirement items that constitute a combination of FRs/CFs and QRs (FR/CF+QR), 12 FRs with constraints over functions (FR+CF), and 8 domain assumptions (DA). Statistically, QRs by themselves account for 51% of the NFRs, and quality-related requirements (QR, FR/CF+QR) account for 67%. Moreover, there are 21 FRs and 36 CFs which we judge to contribute to QRs (e.g., security-related CFs contribute to security), bringing up the total of QR-related requirements to 82% in the sample dataset. These statistics support the claim that most NFRs are indeed quality-related [24], and support the need for an explicit classification of function constraints (CFs).

TABLE II. STATISTICS OF THE CLASSIFICATION OF THE 370 NFRS

| NFR Category | Org. | QR | FR/CF + QR | FR | CF | FR + CF | DA |
|-----------------|------|-----|------------|--------|--------|---------|----|
| Usability | 67 | 47 | 13+1 | 5(3) | 1(1) | 0 | 0 |
| Security | 66 | 2 | 11+3 | 14(11) | 32(32) | 4 | 0 |
| Operational | 62 | 11 | 10+2 | 14 | 12(3) | 6 | 7 |
| Performance | 54 | 44 | 4+1 | 3(2) | 1 | 1 | 0 |
| Look and Feel | 38 | 20 | 7+2 | 9(1) | 0 | 0 | 0 |
| Availability | 21 | 21 | 0 | 0 | 0 | 0 | 0 |
| Scalability | 21 | 19 | 0 | 1 | 0 | 1 | 0 |
| Maintainability | 17 | 8 | 5 | 0 | 4 | 0 | 0 |
| Legal | 13 | 11 | 0 | 2(2) | 0 | 0 | 0 |
| Fault tolerance | 10 | 4 | 2 | 4(2) | 0 | 0 | 0 |
| Portability | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Total | 370 | 187 | 61 | 52(21) | 50(36) | 12 | 8 |

Org.: original categorization; QR: quality requirements; FR: functional requirements; CF: constraints over functions; FR + CF: the combination of FR and CF; FR/CF+QR: the combination of FR and QR, or CF and QR; DA: domain assumptions. Interested readers can find the original data of our evaluation at <http://goo.gl/8ALJDq>.

Examining the data more closely, we find that 151 of the 187 QRs (81%) are classified under usability, performance, availability, look and feel, and scalability; 28 out of 52 FRs (54%) are classified under security and operational requirements; and 32 out of the 50 CFs (64%) are security-related. Moreover, our analysis indicates that the majority of security requirements are, in fact, functional or constraints over functions. E.g., “*The website shall prevent the input of malicious data*”, originally labeled as a security NFR, should actually be a functional requirement. Our dataset includes many requirements of the form “*only users with <role> are allowed to perform <action> or access <asset>*”. In our classification, these were treated as constraints over functions (CFs), not NFRs, since the system-to-be is required to check whether an actor is authorized to act on an asset.

We analyzed the 248 quality-related NFRs (187 QRs and 61 FR/CF+QRs), and identified 67 unique qualities with 327 occurrences (i.e., 327 QGs). The most frequent ones are operability, availability, processing/response time, and scalability.

TABLE III. THE STATISTICS OF SATISFACTION TYPES AND IMPLICIT PRESENCE OF THE OPERATORS ON THE PROMISE NFRS

| Satisfaction Type | NFRs# | Operator Stats. | NFRs# |
|-------------------|-------|------------------|-------|
| Ambiguous | 5 | Universality (U) | 50 |
| Unsatisfiable | 86 | Gradability (G) | 10 |
| Vague | 143 | Agreement (A) | 16 |
| Measurable | 333 | | |

Our analysis of the 370 NFRs resulted in 481 requirements statements using our framework. These were further classified, as shown in the first two columns of Table III. Note that a statement can be tagged with more than one type, e.g., “*all users shall be authenticated*” is practically unsatisfiable, but also measurable, thus the sum of this classification is greater than 481 (in fact, 567). This classification found 15% (86/567) of the statements to be practically unsatisfiable, 25% (143/567) vague and only 59% (333/567) measurable.

We analyzed the implicit application of our three operators **U**, **G**, and **A** to the 481 requirements statements. E.g., “*80% of the users report the user interface is simple*” captures agreement. We show these counts in the last two columns of Table III: 50

U (i.e., stating percentages), 10 **G**, and 16 **A**. Our analysis shows that few of the statements have (implicitly) used **U** and **A**. Particularly, **G** is rarely used. Meanwhile, we found that many NFRs, which need to be relaxed to become satisfiable and measurable, have not been adequately dealt with.

Among the 481 statements, 86 of them implicitly or explicitly use universal quantifiers, e.g., *all*, *any* and *each*, (counted as unsatisfiable in Table III) and likely need to be relaxed using **U** to be properly treated. Also, 36 subjective statements are identified, e.g., *look*, *readability*, *usefulness*, etc., indicating that at least another 20 requirements should be relaxed using **A**. Lastly, **G** could be applied to unsatisfiable (e.g., *almost all*), vague or measurable requirements (see the discussion of **G** in Section IV.B), thus all the 476 items (except the 5 ambiguous ones) are candidates for the **G** operator, but only 10 actually (implicitly) used it (e.g., *fast enough*). E.g., “*the interface shall be appealing*”, as found in the dataset, is clearly gradable.

B. Using our Proposed Methodology: Meeting Scheduling

We use *meeting scheduler*, one of fifteen projects in the PROMISE requirements dataset, to evaluate the expressiveness of our framework and illustrate how our goal modeling language and methodology can be applied to a realistic case study. Functionally, the *meeting scheduler* is required to create meetings, send invitations, book conference rooms, book room equipment and so on. This example includes 47 NFRs, covering different aspects of the system, such as interoperability, usability, security, user friendliness, etc.

We analyzed the 47 NFRs, and identified 21 QRs, 9 FRs, 14 FR+QR, 2 CF+QR, and 1 DA. We captured the 37 items (excluding the 1 DA and 9 FRs) using our framework, resulting in 58 quality goals (an NFR may refer to more than one quality), concerning 27 unique qualities. Frequently used qualities include interoperability, operability, scalability (concurrent capacity), etc. We managed to rewrite all the 58 QGs using our syntax, validating the expressiveness of our framework. In this project, we did not find ambiguous NFRs.

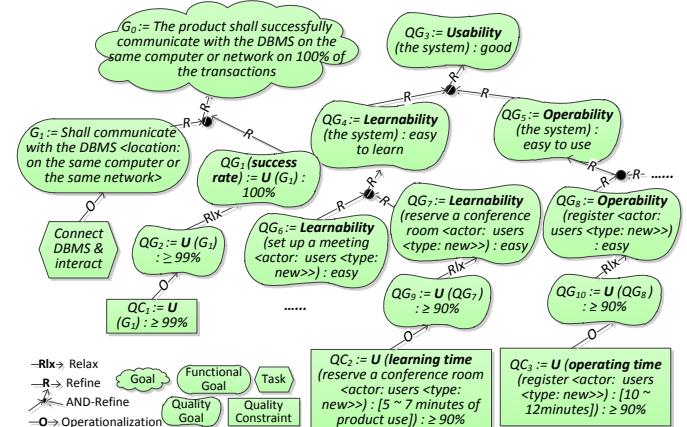


Fig. 3. The NFR model of meeting scheduler (partial)

For space reasons, we only show part of the goal model with a subset of the goal modeling elements in Fig. 3 (note that *focus* is a sub-type of *refine*, and we do not distinguish between them in the model). The full model can be found online:

<http://goo.gl/AxNjPf> (it has more than 58 QGs because it also includes their refinements). In our model, quality goals are formally captured using our framework rather than informal descriptions stated in natural language. Moreover, these were systematically refined and operationalized into quality constraints by following our methodology, making the informal quality goals satisfiable, and measurable.

C. Summary

We have evaluated the need for our framework using real data. The statistical analysis tells us that quality plays a key role among NFRs in RE practice, and that many requirements identified as NFRs are actually constraints over FRs. These results help to justify our classifications of requirements (Fig. 2). Moreover, many NFRs are ambiguous, (practically) unsatisfiable, vague, and subjective, demonstrating the need for the meta-qualities as introduced in our language. Lastly, we have tested the expressiveness of our framework by capturing the NFRs of *meeting scheduler* using our syntax, and illustrated our methodology by performing an in-depth case study. Our extensive results show that the framework is adequate for covering NFRs in practice, and, in fact, could improve RE practice by prompting for the elicitation of unambiguous, satisfiable and measurable NFRs.

VII. RELATED WORK

A. Definitions of NFRs

On the key topic of what are NFRs and how to deal with them, there have been many efforts. Notably, there have been two recent reviews by Martin Glinz [25] and Chung et al. [26].

Glinz [25] surveys thirteen NFR definitions and suggests his own, based on attributes and constraints. However, his definition focuses on only system NFRs, and does not take into consideration project and process requirements (e.g., development, deployment, maintenance, etc.). In our framework, such processes will serve as subjects, allowing us to specify different kinds of desired qualities of them. Moreover, his proposal does not offer methodological guidance for designing language and/or method for capturing NFRs.

After analyzing a list of NFR definitions, Chung et al. [26] define functional requirements as mathematical functions of the form $f: I \rightarrow O$ (I and O represent the input and output of f), and NFRs are anything that concern characteristics of f , I , O or relationships between I and O . This definition, accordingly, will treat constraints over functions (CFs) as NFRs, e.g., “*updates of the database can only be performed by managers*” is a function constraint in our framework (it requires the system to check who is updating, and is not about the quality of updating), but an NFR according to Chung et al. [26]’s definition.

B. The Treatment of NFRs

General approaches. In general, NFRs are classified into sub-categories and represented by plain or structured sentences. E.g., the IEEE standard 830 [27] classifies NFRs into interface requirements, performance requirements, attributes and design constraints, and document them in a separate section from FRs using plain English. The Volere template [28] presents a struc-

ture for requirements sentences, including ID, type, description, rationale, dependencies, etc., all of which are informal and textual information. Moreover, such approaches classify requirements strictly as function/non-functional, and do not support capturing requirements which combine functional and quality aspects. Our evaluation, as well as the work of Svensson et al. [29], has found many such examples in practice. These approaches also do not reflect the cross-cutting concern of NFRs (i.e., a quality can have multiple parts of a system as its subject).

NFRs have been also used along with structural requirements representation languages like UML use case and class diagram, which are widely used to capture FRs in industry [30] [31]. These approaches closely relate NFRs with FRs, associating NFRs with use cases or mapping the operationalizations of NFRs to operations/attributes of UML classes. By relying on the NFR framework [3] to refine and operationalize NFRs, they are able to capture the gradability, but do not take into consideration the universality and agreement of NFRs.

Goal-oriented approaches. Goal-oriented approaches are the first to treat NFRs in depth among many approaches [26]. The NFR framework [2][3] was the first to use vague *softgoals* to capture NFRs. In line with this position, many efforts have been devoted to goal models, resulting in many frameworks, such as *i** [10], Tropos [32], and Techne [11].

The NFR framework [3] has used *type* and *topic*, which are similar to *quality* and *subject*, to represent softgoals (e.g. “*accuracy [account]*”, wherein “*accuracy*” is a type and “*account*” is a topic). However, it did not further explore qualities (types), subjects (topics) and quality values or spaces, which could give rise to the universality, gradability and agreement of NFRs. Similarly, Jureta et al. [33] have used DOLCE quality to distinguish between NFRs and FRs, and define *softgoal* and *quality constraint*, but they did not focus on the ontological meaning of quality, i.e., what kind of subjects a quality can inhere in within a software system, what is the structure of its value space, what kind of quality value it has (vague vs. crisp), etc.

Earlier work by Jureta et al. [34] has identified similar issues in treating NFRs as softgoals (e.g., subjectivity, imprecision, idealism). They deal with these issues by precisely defining softgoals using templates, emphasizing quantification and eventual formalization of refinements. Softgoals in their approach are de-idealized by finding specific targets derived from benchmarks, while our framework offers a richer set of operators for relaxing quality goals. Unlike our work, their proposal has not yet been validated through application to real data.

Note that we are not the first to use the concept “*quality goal*”. Lamsweerde [35] has already proposed this term, but simply treated it as a softgoal.

Quality-oriented approaches. Quality is the most popular term adopted to specify NFRs. Many efforts have been made towards classifying and quantifying qualities, resulting in fruitful quality models and techniques. The famous models include Boehm et al. [36], ISO/IEC 9126-1 [16], ISO/IEC 25010 [1], etc., in which qualities and their interdependencies are usually organized in a hierarchical structure.

One issue with these quality models is that they, even the well-known ones, are neither terminologically nor categorically

consistent with each other [26]. E.g., “*understandability*” is a sub-quality of “*usability*” in ISO/IEC 9126-1 [16], but is a sub-class of “*Maintainability*” in Boehm et al. [36]. In our proposal, the categorization of qualities (or NFRs) will be transformed into two open questions: what kind of subjects needs to exist? (e.g., artifacts, functions or processes) and what kind of qualities will inhere in them? Our answer to them is to develop ontologies containing a number of upper-level categories showing what the qualities and subjects can be and let stakeholders to decide depending on the system domain.

Quality quantification is another important aspect on dealing with quality requirements. It is similar to our focus refinement and operationalization: a quality is often decomposed to several sub-qualities and then quantified using metrics [21][38]. One can refer to the example of “*learnability*” adopted from ISO/ IEC 9126-2 [21] and discussed in section V.B.

The planning language (i.e., Planguage), proposed by Tom Glib [38] and widely used in industry, is a typical example along this line: it uses a set of keywords such as *scale*, *meter*, *must* and *plan*, and a *syntax* that captures *fuzzy concepts*, *quantifiers* and *collections* to express and quantify quality requirements. However, Planguage is informal and textual, and does not allow compositional notions for specifying the subjects of concerned qualities as well as the degree of fulfillment of other requirements. Moreover, it does not offer a methodology for refining informal stakeholder goals to unambiguous, satisfiable and measurable requirements, only providing a language to capture the results of such a process.

Uncertainty and vagueness. These two characteristics of requirements have been actively discussed in RE [19][39][40].

Statistical uncertainty has been extensively studied in goal models, e.g., KAOS [39] and Tropos [41]. These approaches use probability theory to propagate the satisfaction of goals in an AND/OR refined hierarchy. Fuzzy logic has been applied to capture the vagueness of requirements for trade-off analysis at early stage [42]. Recently, the vagueness of NFRs has been explored for adaptation [19][40].

Although probability theory and fuzzy logic have been adopted to handle uncertainty and vagueness, they are often not well distinguished: *probabilistic*, which indicates the probability that a requirement can be true or false, is usually confused with *fuzzy*, which implies the degree to which a requirement can be satisfied.

Our proposal captures these two features by using the **U** and **G** operators. Moreover, our framework allows the combination of these features, which is very practical: either fuzzy probability (e.g., *nearly 90% of the time*) or probability over fuzzy (e.g., *nearly 5 sec. 90% of the time*).

Summary. We summarize the related work with regard to the three key features of NFRs, **U**, **G**, and **A**, in Table IV, wherein ‘ \checkmark ’ and ‘ \times ’ means supported and unsupported, respectively. One should distinguish between vague and gradable: vague requirements are requirements with indeterminate satisfaction conditions (e.g., *high availability*, *low cost*, and *fast response*) while gradable means the degree of fulfillment of a requirement can be graded, resembling the fuzzy membership in Fuzzy Logic. Please also note that although agreement has

been recognized as an important aspect in RE [43] since the early ‘90s, few of the surveyed approaches have captured it.

TABLE IV. AN OVEWVIEW ON COMPARING RELATED APPRAOCHES

| App. | Source | Technique | T | U | G | A | C |
|----------------------|----------------------|--------------------------|----------|--------------|--------------|--------------|--------------|
| General | IEEE 830 [27] | English | N | \checkmark | \checkmark | \checkmark | \checkmark |
| | Robertson et al.[28] | Volere | N* | \checkmark | \checkmark | \checkmark | \checkmark |
| | Supakkul et al.[30] | Use case | S | \times | \times | \times | \times |
| | Cysneiros et al.[31] | Class diag. | S | \times | \times | \times | \times |
| | Whittle et al. [40] | Fuzzy logic | F | \times | \checkmark | \times | \times |
| | Yen et al. [42] | Fuzzy logic | F | \times | \checkmark | \times | \times |
| Goal Oriented (GORE) | NFR framework[3] | Goal model | S | \times | \checkmark | \times | \times |
| | KAOS [9][39] | Probability | F | \checkmark | \times | \times | \times |
| | i* [10] | i* | S | \times | \checkmark | \times | \times |
| | Tropos [32][41] | Probability; Fuzzy logic | F | \checkmark | \checkmark | \times | \times |
| | Techne [11][33] | Techne | S | \times | \checkmark | \times | \times |
| | Jureta et al. [34] | Goal model | S | \times | \checkmark | \times | \times |
| Quality Oriented | Tom Glib [38] | Planguage | N* | \checkmark | \checkmark | \times | \times |
| GORE+ Quality | Our Work | Ontology | F | \checkmark | \checkmark | \checkmark | \checkmark |

App.: approach; T: type; N: informal (plain English); N*: informal (structured English); S: semi-formal; F: formal; U: universality; G: gradability; A: agreement; C: compositional

There have been some discussions about universality (**U**). Berry et al. [44] argues that the use of “*all*” in requirements specification documents expressed in natural language is “dangerous” and practically unfulfillable when used to define domain assumptions, but a “*laudable goal*” when used to describe requirements. We are arguing the opposite in our work. Use of universals in a requirement is dangerous, meaning, fulfillment of the requirement is unrealizable or at least expensive. Use of universals in a domain assumption is fine because it simply states that the design will work only if the assumption holds. That is, such use of universals simply (and usefully) delimits the scope of the solution represented by the design.

VIII. CONCLUSION

In this paper, we treat NFRs as requirements over qualities, proposing a compositional language and a goal-oriented methodology to capture them. We start with quality mappings, and then discuss the domain and codomain of qualities: a collective subject can lead to **universality**, a subjective one likely needs **agreement**, and quality values versus desired quality regions will give rise to the **gradability** of NFRs [14]. We propose a goal-oriented methodology to refine early and informal NFRs to make them unambiguous, satisfiable and measurable. Our proposal serves to: (1) better understand what NFRs are; (2) better distinguish between NFRs and FRs; (3) write better NFR specifications.

Several issues remain open. One is the integration of contribution links with our formalism. As we have discussed, contribution links are indispensable because they capture the influences of functions or function constraints on quality goals. However, integration with our quality-based formalism has not yet been fully explored.

Another important issue is how to perform reasoning on our revisited goal models. In our framework, the satisfaction of a quality goal will depend on the membership between the measured quality value and the expected quality region. Moreover,

when a quality is refined to several sub-qualities, its co-domain will be a multi-dimensional space with each of its sub-quality as a dimension. As such, the reasoning over quality goal satisfaction in our framework will differ from classical goal model reasoning (e.g., [45]), and needs to be further investigated.

ACKNOWLEDGMENT

We are grateful to the anonymous reviewers for helpful suggestions that enabled us to much improve this paper in content and presentation. This research has been funded by the ERC advanced grant 267856 “Lucretius: Foundations for Software Evolution”, unfolding during the period of April 2011 - March 2016. It is also financially supported by the National Natural Science Foundation of China (No. 61033006).

REFERENCES

- [1] ISO/IEC 25010, “Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models,” 2011.
- [2] J. Mylopoulos, L. Chung, and B. Nixon, “Representing and using nonfunctional requirements: A process-oriented approach,” *Software Engineering, IEEE Transactions on*, vol. 18, no. 6, pp. 483–497, 1992.
- [3] L. Chung, B. A. Nixon, and E. Yu, *Non-Functional Requirements in Software Engineering*, vol. 5. Kluwer Academic Pub, 2000.
- [4] F.-L. Li, J. Horkoff, J. Mylopoulos, L. Liu, and A. Borgida, “Non-Functional Requirements Revisited,” *iStar*, 2013, pp. 109–114.
- [5] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari, “Ontology Library,” *WonderWeb Deliverable D18*, 2003.
- [6] G. Guizzardi, *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology, 2005.
- [7] T. Menzies, B. Caglayan, H. Zhimin, K. Ekrem, K. Joe, P. Fayola, and T. Burak, “The PROMISE Repository of empirical software engineering data,” Jun-2012. [Online].Available: <http://promisedata.googlecode.com>.
- [8] “Feature structure,” Wikipedia, the free encyclopedia. 16-Jan-2014.
- [9] A. Dardenne, A. Van Lamsweerde, and S. Fickas, “Goal-directed requirements acquisition,” *Science of computer programming*, vol. 20, no. 1–2, pp. 3–50, 1993.
- [10] E. S.-K. Yu, “MODELLING STRATEGIC RELATIONSHIPS FOR PROCESS REENGINEERING,” University of Toronto, 1995.
- [11] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, “Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling,” *RE*, 2010, pp. 115–124.
- [12] F. Dalpiaz, V. Silva Souza, and J. Mylopoulos, “The Many Faces of Operationalization in Goal-Oriented Requirements Engineering,” *APCCM*, 2014.
- [13] P. Gärdenfors, *Conceptual spaces: The geometry of thought*. MIT press, 2004.
- [14] R. S. S. Guizzardi, F.-L. Li, A. Borgida, G. Guizzardi, J. Horkoff, and J. Mylopoulos, “An Ontological Interpretation of Non-Functional Requirements,” *FOIS*, 2014.
- [15] A. Albuquerque and G. Guizzardi, “An ontological foundation for conceptual modeling datatypes based on semantic reference spaces,” *RCIS*, 2013, pp. 1–12.
- [16] ISO/IEC 9126-1, “Software Engineering - Product Quality - Part 1: Quality Model,” 2001.
- [17] ISO/IEC 25030:2007, “Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality requirements.”
- [18] R. J. Brachman and H. J. Levesque, *Knowledge representation and reasoning*. Morgan Kaufmann, 2004.
- [19] L. Baresi, L. Pasquale, and P. Spoletini, “Fuzzy goals for requirements-driven adaptation,” *RE*, 2010, pp. 125–134.
- [20] I. J. Jureta, S. Faulkner, and P.-Y. Schobbens, “Clear justification of modeling decisions for goal-oriented requirements engineering,” *Requirements Engineering*, vol. 13, no. 2, pp. 87–115, 2008.
- [21] ISO/IEC 9126-2, “Software engineering - Product quality - Part 2: External metrics,” International Organization for Standardization, Geneva, Switzerland, 2003.
- [22] P. Gärdenfors, *Meanings as conceptual structures*. Lund University, 1995.
- [23] G. Elahi and E. Yu, “Modeling and analysis of security trade-offs–A goal oriented approach,” *Data & Knowledge Engineering*, vol. 68, no. 7, pp. 579–598, 2009.
- [24] B. Paech and D. Kerkow, “Non-functional requirements engineering–quality is essential,” *REFSQ*, 2004.
- [25] M. Glinz, “On non-functional requirements,” *RE*, 2007, pp. 21–26.
- [26] L. Chung and J. do Prado Leite, “On non-functional requirements in software engineering,” in *Conceptual Modeling: Foundations and Applications*, 2009, pp. 363–379.
- [27] I. C. S. S. E. S. Committee, I. Electronics Engineers, and I.-S. S. Board, *IEEE recommended practice for software requirements specifications: approved 25 June 1998*, vol. 830. IEEE, 1998.
- [28] J. Robertson and S. Robertson, “Volere: Requirements specification template,” *Technical Report Edition 6.1*, Atlantic Systems Guild, 2000.
- [29] R. Berntsson Svensson, T. Olsson, and B. Regnell, “An investigation of how quality requirements are specified in industrial practice,” *Information and Software Technology*, vol. 55, no. 7, pp. 1224–1236, 2013.
- [30] S. Supakkul and L. Chung, “Integrating FRs and NFRs: A use case and goal driven approach,” *framework*, vol. 6, p. 7, 2005.
- [31] L. M. Cysneiros and J. C. S. do Prado Leite, “Using UML to reflect non-functional requirements,” *CASCON*, 2001.
- [32] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, “Tropos: An agent-oriented software development methodology,” *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.
- [33] I. J. Jureta, J. Mylopoulos, and S. Faulkner, “Revisiting the core ontology and problem in requirements engineering,” *RE*, 2008, pp. 71–80.
- [34] I. Jureta, S. Faulkner, and P. Y. Schobbens, “A more expressive softgoal conceptualization for quality requirements analysis,” *ER*, 2006, pp. 281–295.
- [35] A. van Lamsweerde, “Goal-oriented requirements engineering: a roundtrip from research to practice [engineering read engineering],” *RE*, 2004, pp. 4–7.
- [36] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, and M. J. Merrit, *Characteristics of software quality*, vol. 1. North-Holland Publishing Company, 1978.
- [37] R. B. Svensson, M. Host, and B. Regnell, “Managing quality requirements: A systematic review,” *SEAA*, 2010, pp. 261–268.
- [38] T. Gilb, *Competitive engineering: a handbook for systems engineering, requirements engineering, and software engineering using Planlanguage*. Butterworth-Heinemann, 2005.
- [39] A. Cailliau and A. van Lamsweerde, “A probabilistic framework for goal-oriented risk analysis,” *RE*, 2012, pp. 201–210.
- [40] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, “Relax: Incorporating uncertainty into the specification of self-adaptive systems,” *RE*, 2009, pp. 79–88.
- [41] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, “Reasoning with goal models,” *ER*, 2002, pp. 167–181, 2003.
- [42] J. Yen and W. A. Tiao, “A systematic tradeoff analysis for conflicting imprecise requirements,” *RE*, 1997, pp. 87–96.
- [43] K. Pohl, “The three dimensions of requirements engineering,” *CAiSE*, 1993, pp. 275–292.
- [44] D. M. Berry and E. Kamsties, “The dangerous ‘all’ in specifications,” *IWSSD*, 2000, pp. 191–193.
- [45] J. Horkoff and E. Yu, “Comparison and evaluation of goal-oriented satisfaction analysis techniques,” *Requirements Engineering*, pp. 1–24, 2012.

Quality Requirements Elicitation Based on Inquiry of Quality-Impact Relationships

Farnaz Fotrousi

Blekinge Institute of Technology
(BTH)
Karlskrona, Sweden
farnaz.fotrousi@bth.se

Samuel A. Fricker

Blekinge Institute of Technology
(BTH)
Karlskrona, Sweden
samuel.fricker@bth.se

Markus Fiedler

Blekinge Institute of Technology
(BTH)
Karlskrona, Sweden
markus.fiedler@bth

Abstract—Quality requirements, an important class of non-functional requirements, are inherently difficult to elicit. Particularly challenging is the definition of good-enough quality. The problem cannot be avoided though, because hitting the right quality level is critical. Too little quality leads to churn for the software product. Excessive quality generates unnecessary cost and drains the resources of the operating platform. To address this problem, we propose to elicit the specific relationships between software quality levels and their impacts for given quality attributes and stakeholders. An understanding of each such relationship can then be used to specify the right level of quality by deciding about acceptable impacts. The quality-impact relationships can be used to design and dimension a software system appropriately and, in a second step, to develop service level agreements that allow re-use of the obtained knowledge of good-enough quality. This paper describes an approach to elicit such quality-impact relationships and to use them for specifying quality requirements. The approach has been applied with user representatives in requirements workshops and used for determining Quality of Service (QoS) requirements based the involved users' Quality of Experience (QoE). The paper describes the approach in detail and reports early experiences from applying the approach.

Index Terms—Requirement elicitation, quality attributes, non-functional requirements, quality of experience (QoE), quality of service (QoS)

I. INTRODUCTION

Quality requirements are an important class of non-functional requirements [1]. They concern software system attributes such as functional suitability, performance, reliability, usability, security, and portability that are important for achieving stakeholder goals [2]. The satisfaction of these quality attributes determines whether the software system meets the goals of its stakeholders or whether the system has a negative impact for these stakeholders [3, 4].

Meeting the right level of quality is important to balance benefits and cost [5]. The quality of a software system needs to be at least as good as to make the software useful and competitive, but should not be excessive to avoid cost and unnecessary use of resources. Insufficient quality leads to disappointment and consequent churn when stakeholders

decide to abandon the software solution and adopt alternatives instead [6]. Excessive quality may lead to an unnecessarily expensive design of the software system [7], to unnecessary consumption of resources needed for operating the system [8], and to trade-offs where other quality attributes suffer [9].

To address the problem of finding the level of good-enough quality, the relationship between software quality and the impacts of such quality for the stakeholders of the software system needs to be understood. As demonstrated for the Quality of Service (QoS) of a telecommunication network and the Quality of Experience (QoE) of the network users, a quality-impact relationship can be developed empirically by setting quality levels of a given quality attribute and measuring the reaction of the stakeholders that were exposed to these quality levels [10].

This paper describes how to use quality-impact analysis for eliciting requirements about good-enough quality of a software system. The proposed method guides the elicitation of the quality-impact relationships and explains how to use the gained insights to specify quality requirements. The method delivers empirical evidence for a specific software system that is more reliable than generic expert opinion. The evidence pertains to the features that were investigated and the stakeholders that were participating in the requirements inquiry, thus is adequate and relevant for decision-making about that software system's quality requirements.

The paper describes the proposed quality-impact elicitation method in depth. It gives details about the key ideas of the method and explains how to tailor the method depending on the investigated quality characteristics, the stakeholder goals impacted by these quality characteristics, and the instruments that the investigator is able to apply. The paper provides an example of how the method is applied in practice by reporting about its use in a real-world software development project.

The paper is structured as follows. Section II reviews existing work and motivates quality requirements elicitation based on quality-impact relationship inquiry. Section III describes the method in-depth. Section IV describes how the method is applied and reports the lessons-learned from such method application. Section V compares the method and the obtained results with related work. Section VI concludes.

II. RELATED WORK

According to ISO/IEC FDIS 25010, the quality of a system is the degree to which the system satisfies the needs of its stakeholders. The determination of whether a system exhibits the desired quality characteristics is not straightforward, however. In contrast to functional requirements, many quality requirements do not have a sharp boundary between satisfaction and non-satisfaction. Instead, they are gradually satisfied [11], thus called soft requirements [12].

The softness characteristic of implies that the right level of desired requirements quality needs to be identified during requirements engineering [5]. Each such quality level has its own specific costs and benefits. High quality levels are considered more costly than low quality levels because more expensive designs or approaches to provision of the software service need to be chosen to implement the requirement. In a similar vein, increase of the quality level implies increase of the benefits generated by the requirement. A product that is considered useless because of too low quality becomes useful or even competitive with increased quality. Too much quality, however, is considered excessive thus not adding any value for stakeholders despite quality improvement. The trade-off between cost and value impacts is a basis to determine the desired quality level and specify the requirement in a quantified manner [13, 14].

Goal models have been proposed to elicit quality requirements [3, 15]. Such models allow identification of needs for improving, increasing, or keeping the level of the quality characteristics of a software. To support systematic identification of goals and qualities within a given domain, ontologies have been developed and used to support requirements elicitation [16, 17]. The means-ends relationships that are an inherent part of a goal model make the impact of such a quality requirements explicit [18, 19]. The goals that are enabled by such a decision are used as a rationale that motivates the quality requirement.

Unfortunately, goal models are of limited help eliciting appropriate levels of quality. Goal models help identifying the quality characteristics that are perceived relevant by stakeholders, and the means-ends relationships connect these qualities to the impact that is desired by the stakeholders. However, they do not guide a requirements engineer in how much of a desired quality is good enough. One of the key limitation is that the goal models do not relate a given quality level to a given level of impact beyond the coarse-grained levels of a requirement being denied, weakly denied, undecided, weakly satisfied, and satisfied [3]. In addition, the application of goal models does not deliver the information needed to quantify a quality requirement, thus make its satisfaction measurable with attributes such as scale and meter [13].

Several supporting elicitation methods have been proposed for requirements elicitation [20]. These include the use of questionnaires, interviews, workshops, creativity methods, storyboards, use cases, role-plays, and prototyping. Review of prototypes has been particularly effective in identifying usability concerns and refining user interaction design to reach

user acceptance [21]. The construction of such prototypes allows a development team to capture assumptions about desired software characteristics and to validate these assumptions, for example by reviewing them as implementation proposals with concerned stakeholders [22, 23].

The supporting elicitation methods provide limited support for the determination of good-enough quality levels because of their generality. Any question can be asked in a questionnaire or interview, any topic explored in a workshop, and a multitude of design decisions be captured with storyboards, use cases, role-play, and prototypes. Guidelines that have been proposed to identify quality requirements [24, 25] target the discovery of quality, but do not help in determining measurable levels of quality. The requirements engineer is thus left with his intuition or experience for asking the right questions [26]. The use of experience, however, is risky as the levels for good-enough quality may change between different software products and product-usage contexts.

To enable requirements engineers to determine appropriate levels of good-enough system quality, we were studying the field of telecommunication. In particular we were looking for approaches that allow the requirements engineer and the system stakeholders understand the meaning of a given level of quality, for example in terms of how the quality level affects the degree of stakeholder satisfaction. In the field of telecommunication, substantial work has been performed for understanding how to measure degrees system quality and how a given degree of system quality affects user attitude [27].

For a telecommunication system, Quality of Service (QoS) requirements are stated that concern system performance, availability, and capacity [17]. Often these requirements are agreed in a service level agreements (SLA) between the system customers and the supplier [28]. User satisfaction is expressed as Quality of Experience (QoE) and refers to the “degree of delight or annoyance of the user of an application or service” [29]. It has been shown that a system’s Quality of Service affects the user’s Quality of experience [27]. Too little user delight and too much user annoyance leads to churn, thus users that try to look for alternatives and try to avoid using the system under consideration.

The knowledge of how QoS is related to QoE has not been translated into requirements engineering methodology yet. In particular, it is unclear how to exploit the relationship between QoS levels with QoE levels in the inquiry of software systems requirements. Also needed is an explanation of how to apply the specifics of the QoS-QoE relationship on the determination of good-enough quality for any system quality attribute and for any important stakeholder need that is impacted by the possible quality levels.

III. QUALITY-IMPACT INQUIRY

This paper proposes a method that we call *Quality-Impact Inquiry* to address the so far unsatisfactorily solved problem of determining adequate levels of quality. As required from a solution proposal, we have explained why a novel method was needed, specify the principles and steps of the method, and

describe how to apply it [30]. To demonstrate that the method is sound, we go a step further than required from a solution paper and report about a preliminary validation that we performed with a real-world software development project. The paper describes the method in sufficient depth to enable replication in practice and further validation research.

The Quality-Impact Inquiry method is based on the principles outlined in our earlier work about the generic relationships between Quality of Service and Quality of Experience [10]. These principles have been translated into a software requirements engineering context by integrating it into an inquiry-based requirements analysis process [31] and combined with prototyping, questionnaires, and workshops as supporting methods for collection of quality measurements and stakeholder opinions. During the workshop, stakeholders are exposed to requirements engineer-defined quality that has been implemented in the prototype and questioned about their perceived quality impact. The correlation between quality measurement and stakeholder opinion is analyzed and used as decision-support to determine and then specify good-enough requirements quality.

The quality-impact inquiry method adapts the inquiry cycle of requirement analysis [31] as follows: the documentation phase is adapted to implement a prototype using a set of accepted requirements described the desired system and collects quality attributes during stakeholder actions. The three elements of requirement discussion phase including questions, answers and reasons are supported by the questionnaire elicitation. Finally the results from the former phases contribute to either freeze or change requirements in the evolution phase.

Fig 1 gives an overview of the Quality-Impact Inquiry process. The remainder of this section describes the generic Quality-Impact Inquiry method and how the method may be tailored. The ensuing section describes how the method has been applied in real-world projects and reports about early lessons-learned.

A. Inquiry Process

Fig 1 gives an overview over the process that characterizes the Quality-Impact Inquiry method. The process contains four steps: preparation, measurement, analysis and decision-making. It is applied iteratively until enough evidence has been collected to decide about what good-enough quality should be for a quality attribute under investigation.

1) *Preparation*: During the first step, Preparation, the materials needed for allowing stakeholders to experience the quality characteristics under investigation are prepared. The work includes the preparation and documentation of a prototype, the formulation of a questionnaire, the recruitment of stakeholders for participation in a workshop, and the scheduling of the workshop.

In the proposed method, quality impact is measured subjectively through a questionnaire. The quality impact is also affected by a real value of quality that is measured objectively [32] and automatically using a prototype. Therefore a list of valid quality requirements are identified from SRS document that is relevant to one feature or a group of features (f) and presented as pairs of quality attribute and value:

$$Q = \{ (q_{att}, q_{val}) | f \} \quad (1)$$

As an example in SRS, a non-functional requirement can be stated as “response time should be less than 2 s”. “Response time” is the attribute and 2 s is the value.

The software might be in a preliminary release (i.e. pre-alpha, alpha and beta testing), a candidate release close to a final product/service, or even a released product ready for an evolution. Preparation of artifacts including a prototype from a software feature(s) and a questionnaire about their quality is the pre-requisite to run the method. The stakeholders experience the software and then answer the questionnaire. Data that are collected from the software use and the questionnaire are analyzed to evolve quality requirements in the software

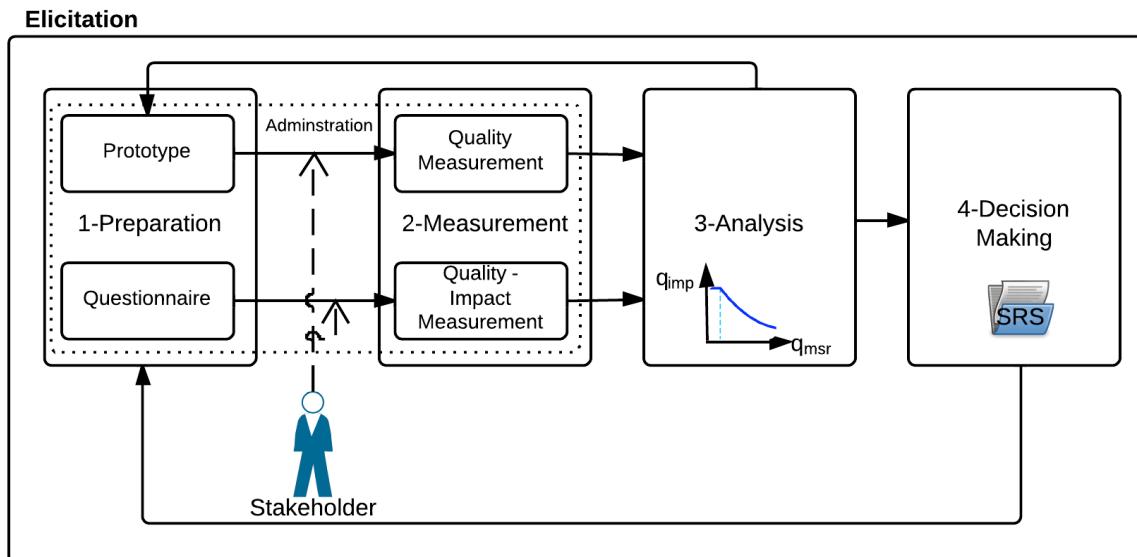


Fig 1. Quality-Impact Inquiry Method

specification document (SRS) if needed.

Based on the quality attributes, the prototype is tailored for the feature(s) f to support measurement of Q . The questionnaire will be tailored using Q to collect quality impacts of feature(s) f relevant to user list U :

$$U = \{u\} \quad (2)$$

Then, scenarios for data collection, and software guidelines to be followed by users are prepared in this step. Translating the questionnaire to the user's mother tongue is another action that might be required.

2) *Measurement*: During the second step, Measurement, a workshop is performed with the aim of collecting quality measurements and user feedback. During the workshop the stakeholders experience predetermined qualities by utilizing the prototype according to a pre-defined script. During the use of the prototype measurements are taken about the quality that the stakeholders experienced. After the use of the prototype, the prepared questionnaire is administered to collect stakeholder opinions about the impacts of the perceived quality.

While the users are using the application through clients such as a smartphone or a PC, quality values q_{msr} (i.e. q_{msr} is a q_{val} relevant to q_{att} for feature(s) f) are quantified by function m , automatically using analytical tools, server log generators or piece of codes embedded in the software.

$$q_{msr} = OP(m(q_{att} | u, f)) \mid OP \in \{MIN \text{ or } MAX\} \quad (3)$$

The function captures the worst value of measured quality attributes in different actions of a user for the given feature(s) f , depending on whether the quality has a success or failure measure characteristic [33]. For a success measure such as availability, the higher value of the quality attribute shows better quality but for a failure measure such as response time, a higher value of the quality attribute shows worst quality. Therefore minimum or maximum value of each case would be the candidate value for measured quality.

Another source of measurement is the questionnaire designed to translate the quality impacts q_{imp} (i.e. q_{imp} is a q_{val} relevant to q_{att} for feature f) into scored values provided by users. In the questionnaire, users are typically asked to provide ratings,

$$q_{imp} = s(q_{att} | u, f) \quad (4)$$

and rationales in forms of comments that explain their ratings:

$$comm = c(q_{att} | u, f) \quad (5)$$

Furthermore, the questionnaire asks users to rate "quality in use" attributes such as satisfaction as a sub list of quality attributes:

$$Q_{inUse} \subset Q \quad (6)$$

The quality impact is translated into a discrete value that is scaled using scores such as Mean Opinion Score (MOS) [34].

3) *Analysis*: During the third step, Analysis, the quality measurements are correlated with the stakeholder opinions about quality impact. This step involves application of statistical analyses based on data that has been collected during the measurement step in the ongoing and previous Quality-Impact Inquiry iterations. The analysis can also be enhanced through a-priori knowledge of the generic nature of the studied quality-impact relationships.

The relation between the measured quality (q_{msr}) and quality impact (q_{imp}) will be identified through a regression analysis, similar to correlation analysis between QoE and QoS [35, 36]. The regression function is calculated for a feature f and quality attribute q_{att} :

$$\hat{q}_{imp}(q_{msr}) = r(q_{msr} | f, q_{att}) \quad (7)$$

Different regression functions for the relationship including linear, logarithmic, exponential and power have potential to be candidate, however the analysis compares the regression function and matches the best one [27].

Then, an estimation of quality value for a given quality impact is calculated by the inverse function of the regression model:

$$q_{msr}(q_{imp}) = r^{-1}(q_{imp} | f, q_{att}) \quad (8)$$

The output of the analysis proposes a list of quality values for different quality impacts including maximum quality impact.

If the quality-impact analysis does not provide enough data for a mature analysis, some changes on the prototype are applied to change the quality values artificially. The looped arrow from analysis box to prototyping box in Fig 1 provides possibilities to achieve enough data for investigating impact changes and perform more reliable analysis than the analysis of less data points.

4) *Decision-Making*: During the fourth step, Decision-Making, the analysis results are used to decide about acceptable and desired levels of quality of the investigated quality attributes. The decisions are recorded in the software requirements specification. The step concludes with decision-making about whether to add inquiry iterations and how the parameters of these ensuing inquiries should be adapted for best improving the knowledge about good-enough quality.

The decision-making process selects suitable quality value from the evidences and decides whether to evolve the value for the relevant quality requirement in the SRS document.

This process identifies maximum applicable quality impact considering technical feasibility, product strategies, and limitation of resources to achieve the relevant quality value, and then applies the decision making function.

Decision-making is a function of parameters including estimated quality value for maximum impact (\hat{q}_{msr}) of a quality attribute, the value of relevant non-functional requirement (q_{SRS}), the list of rationale for the quality attribute rating ($comm$) beside all quality-in-use ratings (Q_{inUse}), to interpret whether the current quality fulfills the users acceptance.

$$q_{new} = \{ g(\hat{q}_{msr}, q_{SRS}, comm, Q_{inUse} | f, q_{att}) \} \quad (9)$$

This function defines a new value for the quality attribute. The decision-making will be performed for all quality attributes in Q .

B. Method Tailoring

There are a wide variety of variation points to adapt the generic Quality-Impact Inquiry process. The variations are needed to be flexible enough to adapt the process to specific requirements engineering constellations. Table 1 gives an overview.

TABLE 1. Estimated quality values for given quality impacts

| Variation Point | Variants |
|------------------------|--|
| Software Features | Stakeholders may be exposed to different features. Quality requirements may be specific to features or the impact of quality levels be perceived differently depending on the feature. |
| Quality Attributes | Stakeholders may be exposed to different quality attributes. Each feature or application may have its own set of prioritized quality attributes. |
| Quality Levels | For the selected quality attributes, different quality levels may be investigated. The selection of the quality level should be based on information need and be guided by statistical analysis methodology. |
| Stakeholder Sampling | Different individuals may be invited for participation in the inquiry workshops. The selected stakeholders should be as representative as possible. |
| Impact Attributes | Stakeholders may be questioned about different quality impacts. Each application or feature may aim at achieving its own specific impacts. |
| Measurements | Different measurements may be selected to record quality levels and stakeholder impacts. |
| Prototyping Approaches | The simulation of different quality characteristics may require different approaches of building the quality-simulating prototype. |
| Impact Function | Different impact functions may be chosen to represent the relationship between a given quality attribute and its impact. We were using linear and exponential functions so far. |

IV. REAL-WORLD EXAMPLE OF METHOD APPLICATION

A. Example Application

To demonstrate how to implement the method in practical situations, we present here the results and lessons-learned of an early validation that we have done in a real-world project. We applied the method for a Diabetes Smartphone Application that will be used by diabetes patients to take blood glucose measurements, to plan insulin injection, and to send the collected observation history to a diabetes specialist for consultation. We evaluated the quality-impact relationships for the features user authentication and observation sharing of diabetes information.

As an input to the quality-impact inquiry we had used a prototype that was instrumented with software for monitoring the timing of user interactions. The inquiry was performed in a laboratory and with a smart phone from the application developers with pre-loaded data. The requirements engineer, the product manager, and selected end-users participated in the inquiry workshop. The inquiry was performed with one end-user at the time.

During the inquiry, the end-user was introduced to the tasks he to be performed with the application, was given a short, tailored user manual, and then used the selected features first according to instructions and then without help. He opened the application, selected the data he wanted to share with his clinician, authenticated himself, and submitted the data. Then the authentication service requested username and password. When authenticated, the data was sent to the application server in the hospital. After the guided and unguided experiences were concluded, the end-user filled out the quality of experience questionnaire. Fig 2 gives an impression of the setup.

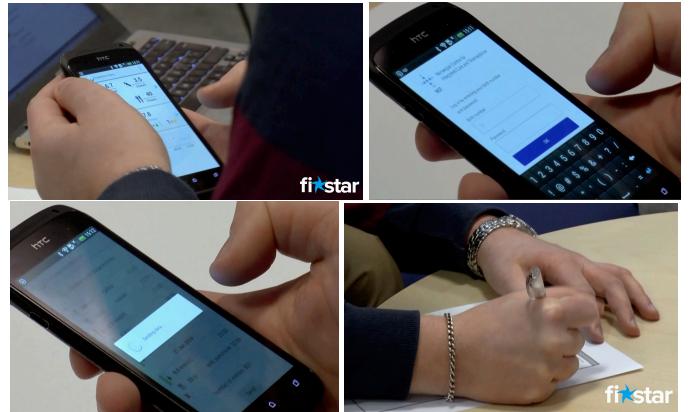


Fig 2: User interaction scenario with instrumented application and subsequent answering of the quality of experience questionnaire.

The quality-impact inquiry processes was implemented for the Diabetes Smartphone Application as follows:

1) *Preparation*: The requirements engineer extracted relevant quality requirements from the software requirement specification document. Based on these extracts he instrumented the software with a time-stamp logger.

The requirements engineer created a short guideline to assist the end-user in using the application. It described the features to be evaluated and how the features should be used.

Based on the extracted quality requirements, the requirements engineer created a quality of experience questionnaire with generic questions about the experience, about the features and product, and about the perceived quality. For the Diabetes Smartphone Application, the quality questions were about performance, reliability, and availability. Fig 3 shows the questionnaire.

| | |
|--|--|
| The Experience | |
| 1. Please tell us the name you would give to the feature: | |
| The Features and Product | |
| 2 Overall, how satisfied are you with the features you just have experienced? | |
| <input type="checkbox"/> Excellent (5) <input type="checkbox"/> Good (4) <input type="checkbox"/> Fair (3) <input type="checkbox"/> Poor (2) <input type="checkbox"/> Bad (1) | |
| Please tell us why you feel that way: | |
| 3. Overall, how good is the feature according to your opinion? | |
| <input type="checkbox"/> Exceptional <input type="checkbox"/> Better than comparable products and features <input type="checkbox"/> Good-enough <input type="checkbox"/> Insufficient | |
| Please tell us why you feel that way: | |
| 4. Will you return to use the product again? | |
| <input type="checkbox"/> Yes <input type="checkbox"/> No | |
| Please tell us why you feel that way: | |
| The Quality | |
| 5. The next question is about response time. With response time we mean the time when you press a button until the software does what it is supposed to do. | |
| How do you rate the response time of the feature? | |
| <input type="checkbox"/> Excellent (5) <input type="checkbox"/> Good (4) <input type="checkbox"/> Fair (3) <input type="checkbox"/> Poor (2) <input type="checkbox"/> Bad (1) | |
| Please tell us why you feel that way: | |

Fig 3: Questionnaire. The last question can be replicated and adapted to any feature the requirements engineer is interested of.

2) *Measurement:* The following steps describe the inquiry workshop that was performed once for each user separately.

In the beginning of the inquiry the requirements engineer welcomed the participants, defined the goals of the inquiry, and shared the agenda of the meeting.

The product manager explained the feature to be used and gave prepared guideline to the end-user.

The end-user used the application according to the instructions. He did so twice to allow us collecting data about the learning and knowledgeable use of the feature. The application generated logs automatically and captured information from the user interaction (see Fig 4 for an example). In all timestamp, the time from the internal clock on smartphone was used. Log entries were created when end-user requests are received and when application screen/data have been displayed. The response time extracted from the example is the duration between two time stamps taken from the starting to the ending of an activity.

```

2014-01-27 15:11:25.029000 fistar.observation_sharing.select_activity.click_button_start_send
2014-01-27 15:11:25.253000 fistar.observation_sharing.send_activity.start_activity
2014-01-27 15:11:25.611000 fistar.observation_sharing.select_activity.stop_activity
2014-01-27 15:11:33.694000 fistar.observation_sharing.send_activity.click_button_start_authorize
2014-01-27 15:11:33.921000 fistar.observation_sharing.send_activity.stop_activity
2014-01-27 15:12:39.978000 fistar.observation_sharing.send_activity.start_activity
2014-01-27 15:12:39.997000 fistar.observation_sharing.send_activity.show_sending_dialog
2014-01-27 15:12:41.787000 fistar.observation_sharing.send_activity.show_send_complete_dialog
2014-01-27 15:12:43.182000 fistar.observation_sharing.select_activity.start_activity
2014-01-27 15:12:43.301000 fistar.observation_sharing.send_activity.stop_activity

```

Fig 4: Extract from the log file with timestamps and activities

After application usage, the requirements engineer provided instructions for answering the quality of experience questionnaire. The user answered the questionnaire accordingly. The answers that were collected with quantitative scales provided data for calculating the quality-impact relationship. The qualitative rationale that the users gave for these values assisted us in interpreting the quantitative values.

At the end of the session, the requirements engineer debriefed the participants and thanked them for the participation.

3) *Analysis:* The filled-in questionnaires and time-stamp logs from all end-users interactions were the inputs for the analysis process. The end-users were satisfied with the quality as they reflected in the questionnaire. Therefore the analysis of this example did not identify any deviation to update quality attributes. However the similar study was conducted in our lab where users perception of response time in downloading a webpage containing an image were collected [37]. The analysis of the data distributions concluded a close match for a regression formula on relations between MOS and response time excluding null opinion scores:

$$\hat{q}_{imp}(q_{msr}) = 4.836 \exp(-0.15 q_{msr}) \quad (10)$$

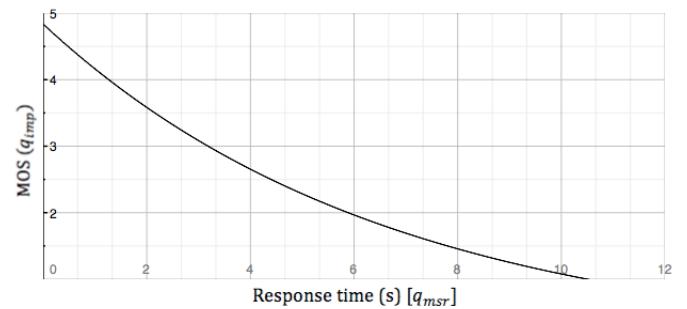


Fig 5. Quality impact (MOS) as a function of quality value (response time(s))

Fig 5 plots this regression function that shows quality impact (q_{imp}) as a function of quality value (q_{msr}) [37]. The response time collected from different experiments as well as collected relevant quality impacts will plot the Fig 4. Taking the reverse of this function estimates quality value (q_{msr}) as a function of quality impact (q_{imp}):

$$\hat{q}_{msr}(q_{imp}) = -6.67 \ln(q_{imp}/4.836) \text{ s} \quad (11)$$

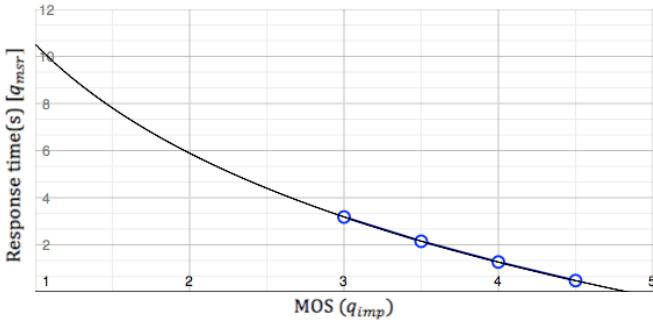


Fig 6. Quality value (Response time (s)) as a function of quality impact (MOS)

As Fig 6 plots the inverse regression function, shorter response identifies the better perception of quality and user's score. Table 2 estimates quality values for q_{imp} in the range of between 3 and 4.5. This value identifies the best threshold value for a quality attribute such as response time that is sufficient for the user expectations.

TABLE 2. Estimated quality values for given quality impacts

| Quality impact (q_{imp}) MOS | Estimated quality value (\hat{q}_{msr}) for Response time |
|----------------------------------|---|
| 4.5 | 0.48 s |
| 4 | 1.27 s |
| 3.5 | 2.15 s |
| 3 | 3.18 s |

4) *Decision making*: Decision making process involves choosing a threshold value for a quality attribute based on inputs from analysis including an estimated quality value for response time, user experiences and rationales, the list of quality-in-use as well as the value of response time defined in the SRS document.

Selecting the good-enough quality level requires trade-offs between the reaching enough user acceptance level instead of maximum level in return for gaining technical feasibility by limited resources such as cost, time and effort. Identifying maximum applicable user perception (quality impact) in each analysis is the result of such trade-offs. If quality impact 4 is recognized enough, then the estimated quality value of 1.27s will be involved in decision making process to update SRS with a good-enough quality value. Typically, the critical value for quality impact is assumed to be 3. In telecommunication area, accepted quality impact in video streaming is considered as 3.5, although the quality impact of 4 is a good choice [38].

B. Lesson learned

1) As shown in the example, the inquiry workshop allowed us to collect the data necessary for analyzing the quality-impact relationship for response time and quality of experience. The workshop lasted about 10 minutes per user. Data aggregation and analysis was concluded within a few hours. Thus the method was relatively efficient. Scalability

can be achieved by working with multiple users in parallel, for example as part of a training workshop.

2) From the users that participated in the inquiry workshops we received positive feedback about the experience and about most of the questions we asked. However, one of the users was puzzled about perceived reliability and availability. He stated that he expected the application to work and to be available in the laboratory situation he was invited to. This shows that usage context affects the relevance of quality attributes. Some quality attributes are relevant in some contexts only. We plan to account for this feedback by extending the quality-impact inquiry to prolonged pilot uses of the application in the real-world contexts of the users.

3) On little usages of the software product could not give the full impression to users. An issue relevant to a quality attribute such as availability might not be risen in a short period of use, this is what reflected by the stakeholder in the example stated in section IV. To reach more accurate data, a prolonged usage should be planned.

4) Not only quality attributes are identified in the proposed quality-impact inquiry method, there might be some proposals for updating functional requirements extractable from the users' comments given in the questionnaire. As an example, if the end-user could not find how to submit the blood glucose data, this could be reflected in the users' perception rating as well as provided rationale.

5) Training before and during the workshop provides knowledge and skills to mitigate the threats of biasing the user perception that occurred due to misuse of the feature. Distractions during the workshop should be removed to boost concentration of users in expressing their real unbiased perception.

V. DISCUSSION

The Quality-Impact Inquiry method is a generic approach to collecting data about quality levels and how these quality levels impact stakeholder satisfaction. It builds on our earlier work that shows that a relationship between quality levels and quality impact can be established. The Quality-Impact Inquiry method extends such earlier work by describing a 4-step process that allows the requirements engineer to inquire how different levels of quality impact the satisfaction of stakeholder needs. The 4-step process is independent of the specific type of quality and independent of the specific kind of stakeholder need. Instead the method can be tailored to any pair of quality and impact measurement that are of interest for the system under consideration. A condition for such tailoring is that a relationship between quality level measurements and impact measurements can be established.

The identified level of quality impact transforms the knowledge into a judgment of good-enough quality. Good enough quality can be decided considering cost and benefit views while exposing barriers and breakpoints [5]. Product strategy decisions, competitors and learning processes are other factors that assist requirement engineer to adjust the level of quality.

The Quality-Impact Inquiry method complements existing quality requirements elicitation methods. Pairs of system quality and impact variables that should be investigated as part of requirements inquiry can be identified with goal-based inquiry methods [3, 15]. Means-ends relationships of prioritized soft goals that relate to system qualities, respectively to stakeholder needs, are candidates for inquiry of the corresponding quality-impact relationships. These candidates are used as an input to the tailoring of the Quality-Impact Inquiry method.

The Quality-Impact Inquiry method utilizes supporting elicitation methods [20], in particular the use of questionnaires, prototypes, and workshops. The method combines these supporting methods into a structured process for creating and analyzing evidence for decision-making about good-enough quality. Recommendations about good practice, e.g. of how to perform an effective workshop [39], should be followed as long as they do not interfere with the objective of the inquiry of quality-impact relationships that are under investigation. Side results from applying the method, e.g. the discovery of new needs or stakeholders during a workshop, should be embraced and handed-over as an input to the main stream of requirements engineering work that is performed in the development project.

In a larger scale validation of the proposed method in a real world situation various stakeholders and experienced requirements engineers are involved. To achieve trustworthy results, a specific probability is identified for considering a confident interval in which the value of quality impact lies within a specific range. Smaller numbers of stakeholders that involve in the experiment method generate wider confidence intervals since there is an inverse square root relationship between the confidence interval and the sample size. It means that to cut error margin in half, number of involved stakeholders is assumed to be four times more.

For practitioners, the Quality-Impact Inquiry method represents an extension of the requirements engineering toolset and is used for addressing the challenging problem of determining good-enough product quality. Once the relevant quality-impact relationships have been established, they can be reused while evolving and maintaining the application and for specifying the quality levels of comparable applications, for example in a software product line.

Quality-Impact Inquiry is not a method that is easy to apply and should thus be used by requirement engineers that are experienced in experimentation with end-users. In many practical situations, this is unproblematic. It is common to use experienced requirements engineers for critical tasks such as the development of service level agreements of software-based services [40].

The Quality-Impact Inquiry method complements competitive analysis of product quality [5]. It allows a definition of thresholds for useful quality and excessive quality based on evidence gathered by analyzing the perception of stakeholders. In the example of QoS and QoE, the requirements engineer determines the service quality threshold by translating quality of experience judgments with the experimentally determined quality-impact relationship. In the real-world

example described in this paper, the former was quantified with software reaction time and the latter expressed with the Mean Opinion Score. The questionnaire in Fig 3 shows that the relationship can also be calculated for other impacts. For example, question 3 was used to collect data about the strategic positioning of the feature according to the Quper model [5]. Question 4 allowed collecting data about the risk of churn. Any prior knowledge about the nature of the relationship, e.g. as expressed by the exponential function in [10], reduces the need for measurements, thus reduces the effort of quality-impact inquiry.

For research, an understanding of the generic relationships between levels of more types of software quality and impact is urgently needed. These generic relationships reduce the need for experimentation during real-world requirements elicitation by pointing to the functions that should be used during quality-impact inquiry. The characterization of the generic relationship between QoS and QoE as an exponential function [10] is an example of the research that is needed. Security and usability are examples of quality attributes that should be prioritized by research. The research may include investigation of what appropriate measurement scales are, e.g. of security or usability, and how a generic quality-impact relationship may be expressed and investigated based on scales other than the ratio scale that we used in Fig 4 and Fig 5. Also open is the development of an understanding of how the interaction of multiple quality variables, e.g. security and usability [9], can be expressed with quality-impact relationships, thus made amenable to requirement elicitation with the Quality-Impact Inquiry method we have presented.

The study of quality-impact relationships would also allow building empirical evidence for checking deeply held beliefs in the requirements engineering field. One such belief is expressed with the KANO model [41]. That model states that the impact of quality on stakeholder satisfaction is expressed through exponential or linear functions that describe attractive requirements, which cause delight when implemented, one-dimensional requirements, which are easily articulated, or must-be requirements, which are not obvious, but considered self-evident by stakeholders. The presented Quality-Impact Inquiry method enables practitioners to determine the exact relationships for the software products and features they are specifying. For researchers, it can be used to inform the design of empirical research studies that aim at investigating generic quality-impact relationships.

VI. SUMMARY AND CONCLUSIONS

The paper has described an approach to quality requirements elicitation based on inquiry of quality-impact relationships. The method, called Quality-Impact Inquiry, guides a requirements engineer in the inquiry of good-enough software quality from the viewpoint of the appropriate stakeholders of the software system. When applying the method, stakeholders experience a prototype of a software system. The requirements engineer collects the real values of chosen quality attributes and subjective feedback from the stakeholders about perceived quality impacts. The analysis of

quality-impact uses a regression function. The method can be tailored to pairs of qualities and impacts that are of interest for the specific software system. Systematic use of the method gives support for deciding about appropriate the quality levels. These can then be specified in a quantified manner for example by stating minimal, maximal, and expected quality in a software requirements specification (SRS) or service level agreement (SLA).

The Quality-Impact Inquiry method was applied for requirements engineering in real-world development projects. One example was shown to describe how to apply the method in practice and to report on lessons-learned. We reported how we have applied the method for these requirements engineering endeavors, shared early experiences from applying the method, and have given recommendations for practical use of the method.

Future research should aim at validating and evaluating the method in further, large-scale requirement engineering situations. Moreover, future research should aim at expanding the understanding of the generic relationships between given combinations of software quality attributes and their impacts as well as how quality attributes interact with each other. The resulting knowledge will translate into a SLA and help to allow and to reuse the knowledge of appropriate quality levels. It will also help accelerating and simplifying quality requirements inquiry in real-world projects, and enable research to check deeply held beliefs about how quality and impacts are interrelated.

VII. ACKNOWLEDGMENTS

This work has been co-sponsored by the European Commission through the FI-PPP integrated project FI-STAR under grant agreement number 318389.

VIII. REFERENCES

- [1] M. Glinz, "On Non-Functional Requirements," presented at the IEEE International Requirements Engineering Conference (RE'07), New Delhi, India, 2007.
- [2] J. Boegh, "A New Standard for Quality Requirements," IEEE Software, vol. 25, pp. 57-63, 2008.
- [3] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering. Boston, USA: Kluwer Academic Publishers, 2000.
- [4] M. Haigh, "Software Quality, Non-Functional Software Requirements and IT-Business Alignment," Software Quality Journal, vol. 18, pp. 361-385, 2010.
- [5] B. Regnell, R. Berntsson Svensson, and S. Olsson, "Supporting Roadmapping of Quality Requirements," IEEE Software, vol. 25, pp. 42-47, 2008.
- [6] K. Kilkki, "Quality of Experience in Communications Ecosystem," Journal of Universal Computer Science, vol. 14, pp. 615-624, 2008.
- [7] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3rd ed.: Addison-Wesley Professional, 2012.
- [8] G. Jung, M. Hiltunen, K. Joshi, R. Schlichting, and C. Pu, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures," presented at the IEEE International Conference on Distributed Computing Systems (ICDCS 2010), Genoa, Italy, 2010.
- [9] C. Braz, A. Seffa, and D. M'Raihi, "Designing a Trade-Off Between Usability and Security: A Metrics-Based Model," presented at the 11th IFIP TC 13 International Conference on Human-Computer Interaction (INTERACT 2007), Rio de Janeiro, Brazil, 2007.
- [10] M. Fiedler, T. Hossfeld, and T.-G. Phuoc, "A Generic Quantitative Relationship between Quality of Experience and Quality of Service," IEEE Network, vol. 24, pp. 36-41, 2010.
- [11] M. Glinz, "Rethinking the Notion of Non-Functional Requirements," presented at the 3rd World Congress for Software Quality, Munich, Germany, 2005.
- [12] C. Irvine and T. Levin, "Quality of Security Service," presented at the 2000 Workshop on New Security Paradigms (NSPW'00), New York, NY, USA, 2000.
- [13] S. Jacobs, "Introducing Measurable Quality Requirements: A Case Study," presented at the 4th IEEE International Symposium on Requirements Engineering (RE'99), Limerick, Ireland, 1999.
- [14] T. Gilb, Competitive Engineering: A Handbood for Systems Engineering, Requirements Engineering, and Software Engineering using Planguage: Butterworth-Heinemann, 2005.
- [15] A. I. Antón and C. Potts, "The use of goals to surface requirements for evolving systems," in International Conference on Software Engineering, Kyoto, Japan, 1998, pp. 157-166.
- [16] A. Souag, C. Salinesi, and I. Wattiau, "Ontologies for Security Requirements: A Literature Survey and Classification," presented at the Advanced Information Systems Engineering Workshops, Gdańsk, Poland, 2012.
- [17] T. Wang, Y. Si, X. Xuan, X. Wang, X. Yang, S. Li, et al., "A QoS ontology cooperated with feature models for non-functional requirements elicitation," in Proceedings of the Second Asia-Pacific Symposium on Internetworks, Suzhou, China, 2010, p. 17.
- [18] L. M. Cysneiros and J. C. Sampaio do Prado Leite, "Nonfunctional requirements: From elicitation to conceptual models," IEEE Transactions on Software Engineering, vol. 30, pp. 328-350, 2004.
- [19] A. Herrmann and B. Paech, "MOQARE: misuse-oriented quality requirements engineering," Requirements Engineering, vol. 13, pp. 73-86, 2008.
- [20] K. Pohl and C. Rupp, Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB Compliant: Rocky Nook Computing, 2011.
- [21] M. Rettig, "Prototyping for Tiny Fingers," Communications of the ACM, vol. 37, pp. 21-27, 1994.
- [22] S. Fricker, T. Gorschek, C. Byman, and A. Schmidle, "Handshaking with Implementation Proposals: Negotiating Requirements Understanding," IEEE Software, vol. 27, pp. 72-80, 2010.
- [23] S. Fricker and M. Glinz, "Comparison of Requirements Hand-Off, Analysis, and Negotiation: Case Study," presented at the 18th IEEE International Requirements Engineering Conference (RE'10), Sydney, Australia, 2010.
- [24] M. Hassenzahl, R. Wessler, and K.-C. Hamborg, "Exploring and understanding product qualities that users desire," in 5th Annual Conference of the Human-Computer Interaction Group of the British Computer Society (IHm-HCI 01), Lille, France, 2001, pp. 95-96.
- [25] R. J. Kusters, R. van Solingen, and J. J. Trienekens, "Identifying embedded software quality: two approaches," Quality and Reliability Engineering International, vol. 15, pp. 485-492, 1999.

- [26] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, "Non-functional requirements in industry-three case studies adopting an experience-based NFR method," in 13th IEEE International Conference on Requirements Engineering, Paris, France, 2005, pp. 373-382.
- [27] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," Network, IEEE, vol. 24, pp. 36-41, 2010.
- [28] H.-B. Kittlaus and P. Clough, Software Product Management and Pricing: Springer, 2009.
- [29] P. Le Callet, S. Möller, and A. Perkis, "Qualinet White Paper on Definitions of Quality of Experience (2012)," European Network on Quality of Experience in Multimedia Systems and Services, Lausanne, Switzerland March 2013 2013.
- [30] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," Requirements Engineering, vol. 11, pp. 102-107, 2006.
- [31] C. Potts, K. Takahashi, and A. I. Antón, "Inquiry-based requirements analysis," IEEE software, vol. 11, pp. 21-32, 1994.
- [32] P. Brooks and B. Hestnes, "User measures of quality of experience: why being objective and quantitative is important," Network, IEEE, vol. 24, pp. 8-13, 2010.
- [33] M. Fiedler and T. Hoßfeld, "Quality of Experience-related differential equations and provisioning-delivery hysteresis," in 21st ITC Specialist Seminar on Multimedia Applications-Traffic, Performance and QoE Miyazaki, Japan, 2010.
- [34] ITU, "ITU-T P.800," in Mean Opinion Score(MOS) terminology, ed: TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 2003.
- [35] H.-J. Kim, D. H. Lee, J. M. Lee, K.-H. Lee, W. Lyu, and S.-G. Choi, "The QoE evaluation method through the QoS-QoE correlation model," in Fourth International Conference on Networked Computing and Advanced Information Management (NCM'08) Gyeongju, Korea, 2008, pp. 719-725.
- [36] T. N. Minhas and M. Fiedler, "Quality of experience hourglass model," in International Conference on Computing, Management and Telecommunications (ComManTel), Ho Chi Minh City, Vietnam, 2013, pp. 87-92.
- [37] J. Shaikh, M. Fiedler, and D. Collange, "Quality of Experience from user and network perspectives," annals of telecommunications-annales des telecommunications, vol. 65, pp. 47-57, 2010.
- [38] A. Khan, L. Sun, E. Jammeh, and E. Ifeachor, "Quality of experience-driven adaptation scheme for video applications over wireless networks," IET communications, vol. 4, pp. 1337-1347, 2010.
- [39] E. Gottesdiener, Requirements by Collaboration: Workshops for Defining Needs: Addison-Wesley Professional, 2002.
- [40] E. Marilly, O. Martinot, H. Papini, and D. Goderis, "Service level agreements: a main challenge for next generation networks," in 2nd European Conference on Universal Multiservice Networks (ECUMN) Colmar, France, 2002, pp. 297-304.
- [41] E. Sauerwein, F. Bailom, K. Matzler, and H. H. Hinterhuber, "The Kano model: How to delight your customers," in International Working Seminar on Production Economics, Igls, Innsbruck, Austria, 1996, pp. 313-327.

Nòmos 3: Reasoning about Regulatory Compliance of Requirements

Silvia Ingolfo
UNITN, Trento Italy
ingolfo@disi.unitn.it

Alberto Siena
FBK, Trento Italy
siena@fbk.it

John Mylopoulos
UNITN, Trento Italy
john.mylopoulos@unitn.it

Abstract—The great impact that law has in the RE-process has called for new techniques and procedures to evaluate the alignment of requirements with applicable laws. In this paper we present a modeling language for the evaluation of compliance of requirements with a piece of law: Nòmos 3. We introduce our language and show the reasoning capabilities of our proposal.

I. INTRODUCTION

Over the past decades, legislators are increasingly laying down new laws to regulate software systems and their use. Requirements engineers are challenged to elicit requirements that satisfy stakeholder needs and also comply with applicable laws. However, regulatory documents are complex artefacts that contain conditional elements such as derogations, exceptions, and so on. Because of this conditional structure, when making design choices, engineers also choose the conditions that the software system will satisfy, and therefore also the clauses that the system should comply to. This, in turn, causes requirements to need to be changed, generating a cost-increasing cycle. For example, when developing a system for a high-school to electronically manage student careers, an analyst may need to take into account privacy laws. A school may be required by law to make public the student final test results, but it is not allowed to disclose mid-term results, except to the single student. In order to produce system requirements that comply with the law, the analyst needs to identify possible ways to satisfy the duty — e.g., publish final results on the school webpage, or send an email with final results to all students — or identify and exploit exceptions to avoid to comply duty — e.g., private schools may not need to make the results public, provided that they officially send the final results to the Education Ministry.

Different solutions exist in RE to support the problem of regulatory compliance of requirements. Many approaches are based on natural language processing techniques, and focus on direct extraction of legal requirements from legal documents [1], [2]. Our approach takes a different perspective and is aimed at helping the analyst to ensure compliance of requirements to a given law by means of conceptual modelling and lightweight reasoning. Laws are modelled using the Nòmos 3 modeling language, and variability points and alternatives are identified through automatic reasoning [3], [4]. The lightweight approach (based on propositional logic) eases the integration with RE practices, and its viability is confirmed by a scalability study [5].

The objective of this paper is to illustrate the reasoning capabilities of Nòmos 3 for evaluating compliance of requirements, and help the requirement analyst answer questions like: given a set of requirements, which norms apply? which norms are violated? how do I comply with a given norm? who is responsible for a norm?

II. NÒMOS 3: THE LANGUAGE

Nòmos 3 [4] is a modeling language for representing legal provisions, such as laws and regulations, and reasoning about the compliance of requirements — represented in terms of Goals [6]. Nòmos 3 models (see figure 1) are built using the following primitive concepts. Situation: a proposition describing a state of affairs of the world, e.g., “It is Christmas Season”. Goal: a particular Situation desired by (at least) a role in the domain, e.g., “Patients data are updated”. Domain Assumption: are particular Situation that hold in the domain, e.g., “The school is private”. Norm: a right/duty that somebody has, e.g. “Duty to not to disclose sensitive information”. Role: somebody who wants a Goal (Social Role, “Doctor”) or who is addressed by a Norm (Legal Role, “Data Processor”). Relationships allow expressing when:

- Situations make a Norm applicable/not applicable (*activate/block* relation), or make a Norm or Goal or other Situation satisfied/not satisfied (*satisfy/break*)
- Goals bring-about Situations (relation *brings-about*)
- Situations can only be brought about by a specific role (*reserved*)
- Social Roles want Goals (*wants*), and play Legal Roles (*play*)
- Legal Roles are in charge of satisfying Norms (*holds*)
- Norms make other Norms applicable, not applicable or complied (*endorse*, *derogate*, *imply*) [3]

Nòmos 3 supports reasoning about a set of requirements represented as goals and their compliance with a given law. The *legal model* represents a law in terms of: (i) the norms it provides, (ii) the conditional elements (exceptions, derogations,...) that introduce *variability* in the way to comply with the norms, and (iii) the *responsibility* of roles associated with norms (hold relationship). Variability is modelled by capturing the situations that make a norm apply or be satisfied, as described in [3]. Responsibility is modelled by capturing the situations that make compliant the role associated with the norm. The *domain model* represents information about requirements, expressed in terms of situations brought-about when goals are achieved, and roles responsible for the goals.

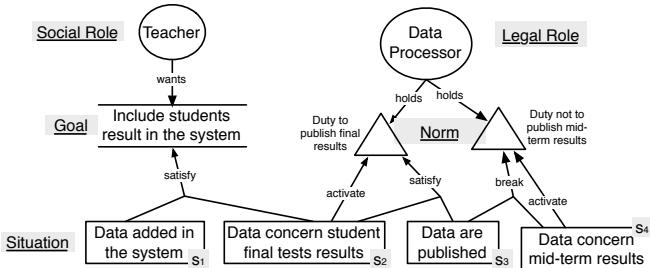


Fig. 1. Example of a Nòmos 3 model

Example. The goal for a Teacher to input the test results in the system (see figure 1), is represented by the Social Role of ‘Teacher’, and the conjunction of two Situations that, when satisfied, make the goal achieved: s_1 “Data are added in the system” and s_2 “Data concern student final-test results”. When the goal is achieved, “Data in the system are updated” is an example of Situation *brought-about* by the Goal. The processing of data concerning students and students results — addressed for example by Title VI of the Italian Privacy laws — addresses the Teacher and the School as “Data Processor”, that who process the data. In the legal model, the Legal Role of ‘Data Processor’ holds the duty to publish the student final-test results, and is responsible to satisfy the norm. The situation s_2 “Data concern student final-test results” identifies what makes this duty applicable; when the situation “Data concern students mid-term results” holds instead of s_2 , then the duty to publish does no longer apply and another duty applies. The duty to publish final results is satisfied when 2 situations hold: s_2 and s_3 “Data are published”. When instead the situations holding are s_3 and “Data concern student mid-term results”, then the duty not to publish mid-term results is not satisfied (it is violated).

III. REASONING WITH NÒMOS 3

Compliance in Nòmos 3 is evaluated by means of inference reasoning. Initial values are assigned to input nodes of the model: Situations which are observed or hypothesized to be satisfied (i.e., the state of affair that they represent holds: ST), false (the state of affair that they represent does not hold: ST) or undefined (it is not known whether the state of affairs that they represent holds or not: SU). Relationship between concepts propagate the values across the legal model in order to perform different types of analysis. A Nòmos 3 model can be queried by means of forward or backward analysis. In forward analysis, input values are simply propagated across the model, and the resulting knowledge is reported (norms complied/violated and roles complying or not). In backward analysis, an explicit query is requested (e.g., specific norms and roles complied), and an assignment satisfying the query is reported, if found. In this analysis, Norms and Roles will receive values through the relations, and from the evaluation of these values a compliance assessment is formulated. In particular, compliance evaluation of norms involves: (a) the identification of applicable Norms (Situations that make a

Norm applicable); (b) satisfied Norms (Situations that make a Norm satisfied); and (c) Legal Roles, who have fulfilled their responsibilities (Situations that should be satisfied by a Role). Compliance evaluation of roles involves the identification of Situations that need to be satisfied by the role: Situations representing the satisfaction of an applicable norm a Legal Role holds, or representing the goal the Social Role wants.

Nòmos 3 models allow the analyst answer questions like:

- *Given a set of requirements, which norm apply?* This question summarizes one of the basic reasoning of Nòmos 3 where a set of Situations are assumed to have a satisfaction value (i.e., goals bring about the sit.), and evaluate which norms these situations make applicable, and the legal roles that should comply with these norms.
- *Given a set of requirements, which norm are complied?* This question summarizes the case where a set of Situations are assumed to have a satisfaction value (i.e., goals bring about the sit.), and evaluate which norms these situations make applicable, satisfied, and make the legal roles comply. Applicable norms not satisfied are violated.
- *Given a set of requirements, which roles in the domain are subject to which norms?* This question summarizes the case where a set of Situations are assumed to have a satisfaction value (i.e., goals bring about the sit.), and evaluate which Social Roles play a Legal Role who is holding an applicable norm.
- *Given a set of requirements, which roles in the domain comply?* This question summarizes the case where a set of Situations are assumed to have a satisfaction value (i.e., goals bring about the sit.), and evaluate the Social Roles that (a) have all their goals satisfied, and (b) the Legal Roles played by the Social Role should have all applicable norms that it holds complied with.

Future work. Currently our model are created manually, so our work-in-progess is working on the semi-automatic generation of Nòmos 3 models from a legal text. Also in future work we plan to implement the primitives of our propositional approach in DLV, to support reasoning over our models.

ACKNOWLEDGMENTS

This work has been supported by the ERC advanced grant 267856 “Lucretius: Foundations for Software Evolution” (April 2011 – March 2016) <http://www.lucretius.eu>.

REFERENCES

- [1] T. Breaux, M. Vail, and A. Anton, “Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations,” in *RE’06*, 2006, pp. 49–58.
- [2] J. Maxwell and A. Anton, “Developing production rule models to aid in acquiring requirements from legal texts,” in *RE’09*, 2009, pp. 101–110.
- [3] A. Siena, I. Jureta, S. Ingolfo, A. Susi, A. Perini, and J. Mylopoulos, “Capturing variability of law with Nòmos 2,” in *ER 2012*, p. 383.
- [4] S. Ingolfo, I. Jureta, A. Siena, A. Perini, A. Susi, and J. Mylopoulos, “Nòmos 3: Legal compliance of roles and requirements,” in *Submitted to ER 2014*, 2014.
- [5] A. Siena, S. Ingolfo, A. Susi, A. Perini, and J. Mylopoulos, “Automated reasoning for regulatory compliance,” in *ER 2013*, 2013.
- [6] E. Yu, “Towards modelling and reasoning support for early-phase requirements engineering,” in *RE’97*, 1997, pp. 226–235.

GUITAR: An Ontology-based Automated Requirements Analysis Tool

Tuong Huan Nguyen, John Grundy, Mohamed Almorsy

Faculty of Science, Engineering and Technology

Swinburne University of Technology

Melbourne, Australia

{huannguyen, jgrund, malmorsy}@swin.edu.au

Abstract—Combining goal-oriented and use case modeling has been proven to be an effective method in requirements elicitation and elaboration. However, current requirements engineering approaches generally lack reliable support for automated analysis of such modeled artifacts. To address this problem, we have developed GUITAR, a tool which delivers automated detection of incorrectness, incompleteness and inconsistency between artifacts. GUITAR is based on our goal-use case integration meta-model and ontologies of domain knowledge and semantics. GUITAR also provides comprehensive explanations for detected problems and can suggest resolution alternatives.

Index Terms—Goal-oriented requirements engineering; Use case; Ontology-based requirements analysis; incorrectness; incompleteness; inconsistency; detection and resolution

I. INTRODUCTION

Goal-use case coupling techniques have been proven to be an effective method in requirements elicitation and elaboration [6]. 3Cs problems (incorrectness, incompleteness and inconsistency) are key challenges of the modeled artifacts' quality. Several techniques have been developed to help in requirements management and analysis. KAOS [7] supports the management of requirements conflicts. However, it does not explicitly detect inconsistency between goals and use cases, incorrectness and incompleteness among the artifacts. Cesar [5] allows the analysis of natural language requirements using domain ontologies. However, its ontological term-mapping technique for 3Cs problem detection may not be sufficient to deal with complicated cases. In addition, it does not support analysis in goal-use case models. RAT/QRA analysis tool [1] depends on adopting linguistic techniques and domain ontologies, which make it limited to linguistic defects (i.e. ambiguity, readability or subjectivity), similarity between pairs of requirements, and unused boilerplates. Thus, there is still a lack of comprehensive automated supports dedicated for goal-use case integration models. For instance, how to verify if a goal or use case is not correctly specified? How to ensure a use case is matched with its associated goal? How to identify whether a required goal/use case has not been elicited? How to identify inconsistent artifact specifications?

To automatically deal with these 3Cs related problems, artifact specifications need to be transformed into a form that is syntactically and semantically understandable by machines. We developed GUITAR (Goal-Use case Integration Tool for Analysis of Requirements) as an extension to our previous

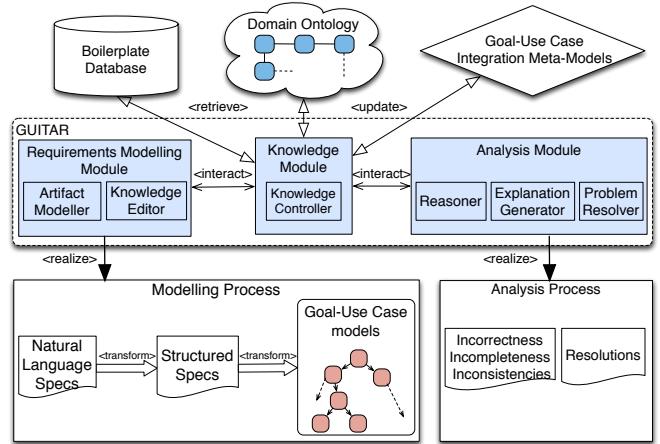


Fig. 1 GUITAR Process

work on inconsistency detection in goal models [3, 4]. GUITAR allows a wide range of natural language artifacts descriptions to be translated into Manchester OWL Syntax and integrated with ontologies of domain knowledge and semantics to automate the analysis process. In addition, our developed meta-model for goal-use case integration enables the detection of syntactical problems. GUITAR also automatically generates explanations and possible resolutions for identified problems.

More information about GUITAR can be obtained from <http://www.it.swin.edu.au/personal/huannguyen/guitar.html>.

II. GUITAR'S PROCESS

Fig. 1 shows an overview of GUITAR. The tool consists of the requirements modeling, knowledge and analysis modules which support the modeling and analysis of requirements.

A. Modelling Process

GUITAR allows the modeling of goals and use cases (referred to as *artifacts* in this paper). These two concepts are integrated based on a meta-model that provides the categorization and valid relationships between them. For each artifact, the tool requires both a natural language specification and a structured specification that is used for automated analysis. In our work, functional grammar [2] is used as the underlining model for structured specifications due to its integration capability with domain ontologies. In this structured specification, each term is mapped to an ontological concept to allow semantic analysis. To help ensure artifact specification's

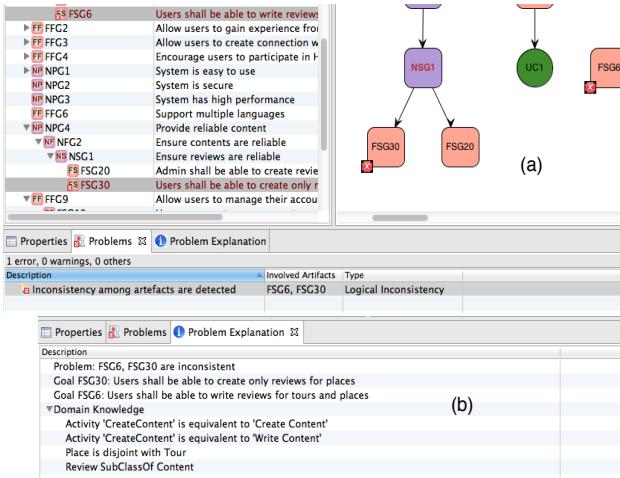


Fig. 2 An Inconsistency Example

quality (i.e. unambiguousness), GUITAR incorporates a database of boilerplates to help writing natural language specifications. For example, if we have a goal “*System shall send update email to users every two weeks*”, a requirements analyst can choose a boilerplate (i.e. $\langle \text{agent} \rangle \text{ shall } \langle \text{action} \rangle \langle \text{object} \rangle \langle \text{beneficiary} \rangle \langle \text{frequency} \rangle$) and fill in the parameters (i.e., $\langle \text{agent} \rangle$). Based on the specified parameters, GUITAR automatically transforms the natural language specification into a structured specification (i.e., $\text{Agent}(\text{System}) + \text{Verb}(\text{Send}) + \text{Object}(\text{UpdateEmail}) + \text{Beneficiary}(\text{User}) + \text{Frequency}(\text{Quantity}(2) + \text{MeasurementUnit}(\text{Week}))$), the terms within parenthesis are ontological concepts). Domain ontologies and boilerplates can be edited (via *knowledge editor* and *knowledge controller* components) and reused in different projects.

B. Analysis Process

GUITAR supports the detection and resolution of 3Cs problems in modeled goals and use cases.

1) Problem Detection - Syntactical problem detection is supported by a 2-layer goal-use case integration meta-model. The artifact layer defines different types of artifacts across levels of abstraction and their relationships while the specification layer provides guidance on the composition of artifacts. The meta-model helps, for instance, with the detection of a missing goal type, an incorrect artifact’s specification, or inconsistent links between artifacts.

GUITAR relies on ontologies of domain knowledge and semantics for semantic requirements analysis. At the core of our technique is the representation of activities. Each activity contains an action and an object (i.e. *create reviews*). Since various relationships between activities may exist in a domain (i.e., “*create content*” requires “*edit content*”), capturing such relations into ontologies is useful to identify mismatches between artifacts (inconsistencies) or missing artifacts (incompleteness). In addition, relationships between concepts or activities (i.e. *user* and *traveller* are equivalent, in a *traveller network* domain, so are ‘*write review*’ and ‘*create review*’ even *write* and *create* are not) provide the semantics of terms used in structured artifact specifications. GUITAR’s *reasoner* allows

structured specifications to be automatically transformed into Manchester OWL Syntax for automated reasoning. Fig. 2 shows an example of a detected inconsistency. The involved artifacts are highlighted (a) and the explanation is provided (b).

2) Problem explanations and resolution suggestions - For detected problems, GUITAR automatically generates resolution alternatives together with detailed problem explanations based on domain ontologies. Explanations are helpful for requirements analysts to get insight into the issues and select appropriate repairing alternatives.

III. EVALUATION

We have evaluated GUITAR with two industrial case studies, one for a traveller social network system and another for an online publishing system. We used precision and recall metrics to assess GUITAR’s soundness and completeness in detecting 3Cs problems. Given domain ontologies of high quality, we have obtained 95% and 90% on average for soundness and completeness respectively.

IV. CONCLUSION AND FUTURE WORK

In this paper, we have presented GUITAR, a tool providing automated analysis of goals and use cases for incompleteness, incorrectness and inconsistencies. GUITAR also enables the generation of problem explanations and resolution alternatives. Our planned future works include: (1) support automated transformation of natural language requirements into goal-use case models. (2) Improve the ontology editor with visualization editing support. (3) Conduct a formal user evaluation on industry practitioners to assess the usability of the tool.

ACKNOWLEDGEMENT

This research is supported by the Victorian Government under the Victorian International Research Scholarships scheme and the Australian Research Council under Linkage Project LP130100201.

REFERENCES

- [1] Requirements Quality Suite. Available from: <http://www.reusecompany.com/requirements-quality-suite>.
- [2] S.C. Dik, “The theory of functional grammar,” Walter de Gruyter, 1989.
- [3] T.H. Nguyen, B.Vo, M. Lumpe, J.C. Grundy, “KBRE: a framework for knowledge-based requirements engineering,” Software Quality Journal, vol. 22, p. 1-33, March 2014.
- [4] T.H. Nguyen, B.Vo, M. Lumpe, J.C. Grundy, “REInDetector: a framework for knowledge-based requirements engineering,” in Proceedings of the 27th International Conference on Automated Software Engineering, 2012.
- [5] A. Rajan and T. Wahl, “CESAR: Cost-efficient Methods and Processes for Safety-relevant Embedded Systems,” Springer, 2013.
- [6] C. Rolland, C. Souveyet, and C.B. Achour, “Guiding goal modeling using scenarios,” IEEE Transactions on Software Engineering, vol. 24, 1998, p. 1055-1071.
- [7] A. V. Lamsweerde, “Goal-oriented requirements engineering: A guided tour,” in Proceedings of Fifth IEEE International Symposium on Requirements Engineering, 2001.

Simulation-Based Requirements Discovery for Smart Driver Assistive Technologies

Andreas Gregoriades

European University Cyprus, Nicosia,
Cyprus

a.gregoriades@euc.ac.cy

Maria Pampaka

The University of Manchester,
UK

maria.pampaka@manchester.ac.uk

Alistair Sutcliffe

The University of Manchester,
UK

Alistair.Sutcliffe@mbs.ac.uk

Abstract—Smart driver assistive technologies (DAT) have been developed to alleviate accident risk by either reducing driver workload or assessing driver attentiveness. Such systems aim to draw drivers' attention on critical cues that improve decision making. However, in some cases, these systems can have a negative effect due to the extra information load they incur to the driver. Therefore, in addition to specifying the functional requirements for such systems there is an urgent need to address the human requirements. This work describes a simulation-based requirements discovery method that utilises the benefits of a modular simulator that models future designs of DAT.

Index terms—Requirements discovery, Simulation, Driver assistive technology

I. INTRODUCTION

Human factors and requirements have a lot to share, however few studies apply human factors knowledge to requirements engineering. While non-functional requirements (NFR) such as performance, security and maintainability are considered for software functions, NFRs for people, such as driver situation awareness (SA) and workload, have received less attention in requirements engineering. Such requirements have been proven very significant in preventing system failure which in traffic safety is articulated in the form of accidents [1]. Therefore, the systematic analysis of this type of NFRs prior to any DAT system implementation is considered vital. The main problem in evaluating these requirements is the need for an implemented hardware-software system prior to a holistic analysis of system behaviour under a number of test scenarios that involve people and tasks. This however is expensive and risky. Therefore, the use of a simulated environment for requirements analysis, saves the costs of physical mock-ups and prototypes, especially for complex systems [2].

This study focuses on the discovery of the requirements of future DAT with emphasis on SA enhancement systems. In traffic safety, SA constitutes a major critical factor, since it provides the driver with the ability to anticipate events given perceived driving and environmental conditions. The study was conducted with local drivers in Cyprus using a driving simulator designed for this purpose. The analysis is based on phenotype driver behaviour data such as lateral deviations, headways, speed, etc. collected through experiments in controlled settings. The method used for driving-behaviour evaluation is based on task-performance. An overview of the simulator and the method used to discover the requirements for future DAT is presented next.

II. DESIGNING THE DRIVING SIMULATOR

The first part of this work involved the design and development of a modular driving simulator that would enable the discovery of human factor requirements of future DAT. The use of a driving simulator is inevitable for such type of analysis for ethical and financial reasons and it allows experimentation under risky conditions. The simulator and its inherent models were, thus, designed to be generic, enabling the utilization of libraries of components representing assets that make up the driving conditions, DAT requirement specifications and road infrastructure characteristics.

The design and development of the driving simulator utilized Unity, City Engine, Maya and Tree[d]. Unity 3D game engine enables the development of 3D computer games and interactive virtual environments; it also enables changing the graphical environment and provides the designer with the ability to define behaviours through a powerful scripting language. City Engine is a 3D procedural modelling software application, specializing in the generation of 3D urban environments through the manipulation of objects and existing GIS data. In our case City Engine was used to manipulate the XML file from OpenStreetMaps and the conversion of its 2D format into 3D using CGA Shape Grammar. Autodesk Maya graphics software was utilized for 3D modelling and animation, and in particular for the creation of vehicles and other static and dynamic models involving various assets (e.g. traffic lights, advertisement billboards). Tree[d] was used for the development of the surrounding environment and vegetation.

Additional car assets have been imported in the simulator through car models, from digital libraries, to provide variety. The selection of the car models was based on car types and brands currently used in Cyprus, in order to enhance the realism factor of the simulator.

The final part of the simulator design was the development of the functionality to enable the interactivity between the user and the simulator. This was realised in Unity through JavaScripting. Upon completion, the user was able to drive the vehicle in the designed environment, by controlling its direction and speed using a steering wheel and pedals.

III. SIMULATION-BASED REQUIREMENTS DISCOVERY METHOD

To discover the requirements of the candidate smart in-vehicle situation awareness enhancement system design, an experiment was conducted using the driving simulator, local

road users and a virtual replica of the road infrastructure in Cyprus. The experiment utilised two combined conditions: roadside advertising billboards with a risky event and in-vehicle music with a risky event, at an accident black-spot in Nicosia (see Figure 1). The human requirement analysed focused on driver's levels of SA and workload. These were assessed based on phenotype driver behaviours, with indicators such as lateral deviations and failure to recall important information of primary task after the experiment.



Fig. 1 The interface of the driving simulator in the virtual road design

Participants drove a pre-specified route in the designed road network both with and without billboards and in-vehicle music at a safety critical point. Data were collected at different stages (i.e. before, during and after the experiment) to capture participants' background information, their engagement and recollection of the conditions, as well as their driving behaviour, workload and SA level indicators.

IV. DESIGN SPECIFICATION FOR A FUTURE SMART SITUATIONAL AWARENESS ENHANCEMENT SYSTEM

Results from the experiments highlighted the need for an SA enhancement system. In particular, results revealed that the highest percentage of accidents occurred at the section where the traffic was increased. This is not surprising since at these locations drivers are expected to process increased volume of information. This highlighted the need of a system functionality that would draw the attention of drivers on safety critical information cues. The criticality of the cues is defined based on proximity to other vehicles, other vehicle trajectories, predicted trajectory of surrounding vehicles and their projected impact. Another important finding is the effect of the vehicle's blind spot on road accidents. Blind spot is the area to the back side of the vehicle where driver's view is obscured by the vehicle's middle pillar. Drivers are taught to look over their shoulder before changing lanes to improve their SA. This finding highlighted the need of additional driver information requirements. The interpretation of what was observed from the experiments, the cause of this behaviour and its translation into system requirements was based on domain knowledge. These focused on issues related to the presentation of critical information to drivers through head-up displays to reduce driver workload and stress.

To respond to these findings and the associated implications, a smart SA enhancement system should support the information needs of drivers by reducing their workload and enhancing their contextual awareness. The challenges in designing such systems are many. The one addressed here is the design of the user interface so that it is not distractive but at the same time informative. This could be achieved by

providing drivers with important situational cues that are critical for maintaining sufficient levels of SA. To reduce the likelihood of accidents, the future system's requirements should focus on information presentation and dynamic risk assessment. To that end the use of an augmented reality overhead display is considered as a suitable option. The visualisations of the display should aim to provide the driver with enhanced peripheral vision with a dynamic assessment of the most critical entities within the immediate periphery of the vehicle. Based on the above, one of the candidate designs proposed and modelled in the virtual environment is depicted in Figure 2. The host vehicle is shown in the middle surrounded by red and green vehicles of different sizes, which, accordingly, correspond to high and low-risk vehicles. Vehicles that are in the driver's blind spot are represented by big red cars. Low risk cars are depicted by small green icons. High proximity or hidden vehicles at intersections are also high risk and hence are big and red. Vehicles positions and speeds can be obtained from on-board sensors. Surrounding vehicles at intersections can be obtained through vehicle-to-vehicle communication protocol. The prototype visualization metaphor presented in Figure 2 will be depicted on the vehicles windshield. This needs to be validated through another experiment to compare safety with/without SA enhancement.

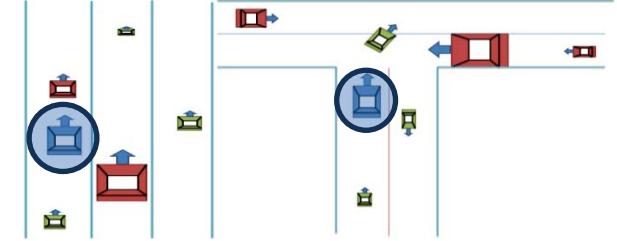


Fig. 2. Visualization of the proposed system design in two traffic scenarios. The host vehicle is shown in a circle.

V. CONCLUSIONS

The simulation-based requirements discovery method enables the design and customization of the road infrastructure for what-if analyses in a modular fashion. This enables the design of the experimental settings for the analysis of requirements under a variety of possible system designs. The use of situation enhancement technology, such as the one specified herein, could alleviate driver overloading problems. The validation of the requirements of the proposed designs, however, needs further experimental simulation studies.

ACKNOWLEDGEMENTS

This work is part of the IPE/NEKYP/0311/02 "VR CAVE" project and is financially supported by the Cyprus Research Promotion Foundation and the European Structural Funds.

REFERENCES

- [1] A. Gregoriades, A. Sutcliffe, "Human-Centred Safety Analysis of Prospective Road Designs," IEEE Transactions on Systems, Man and Cybernetics, Part A, vol. 40, pp. 236-250, 2010.
- [2] B. Gault , A. Sutcliffe, N. Maiden, "ISRE: immersive scenario-based requirements engineering with virtual prototypes", Requirements Engineering 10(2): 95-111 2000

EAM: Ecosystemability Assessment Method

Eric Knauss and Imed Hammouda
Department of Computer Science and Engineering
Chalmers | University of Gothenburg, Sweden
{eric.knauss,imed.hammouda}@cse.gu.se

Abstract—In this extended abstract, we present the ecosystemability assessment method as a means to assess the extent to which a software system, represented by its architecture and its development environment, supports the vision of ecosystem.

Index Terms—software ecosystem, architecture

I. INTRODUCTION

A *software ecosystem* (SECO) has been defined as a set of businesses functioning as a unit and interacting with a shared market for software and services, together with relationships among them [1]. These relationships are frequently underpinned by a common technological platform and operate through the exchange of information, resources, and artifacts [1]. The technological platform is often represented by a software system enjoying various levels of openness. We thus define *ecosystemability* as the extent to which a software system, represented by its architecture and its development environment, supports the vision of ecosystem. For companies, evolving an ecosystem's technological platform represents a complex process, as it needs deep understanding of the SECO phenomenon and the impact of its various facets on platform design decisions. Furthermore, since the SECO trend is relatively recent, there are little technical guidelines on how to grow and maintain a sustainable ecosystem.

In this paper we present ongoing work to design a method for assessing the ecosystemability of a technological platform as well as preliminary results. This method will allow to i) analyze business requirements and architectural drivers in SECOs and ii) to derive and assess architectural candidate solutions for technological platform and development environment. The research questions we would like to explore in this research include the following:

- What constitutes a quality model for ecosystemability?
- What kind of evaluation criteria could be used to assess the ecosystemability of a technological platform?
- How should the assessment be planned and which ecosystem actors should be involved?
- How to obtain data for the assessment process?
- How to exploit the results of the assessment process?

The closest works to our study are existing architecture-centric assessment methods for software systems such as ATAM [2]. These assessment methods, however, focus on analyzing specific software artifacts and quality requirements. What is needed is a review framework that addresses the various dimensions and needs of SECOs. Examples of such dimensions include technical qualities (e.g. architecture has

been adequately designed and documented so that others can contribute to the SECO), licensing issues (e.g. the architecture enables integration of different third party and proprietary components), and the community perspective (e.g. the development environment supports knowledge flow and feedback between ecosystem stakeholders).

EAM (the envisioned Ecosystemability Assessment Method) extends beyond ATAM with respect to the following:

- *Input artifacts*: In addition to architecture design documents, the method considers artifacts, which are used to communicate with other SECO actors, e.g. partnership documents, interfaces, licenses, usage data, etc.
- *Quality attributes*: In addition to quality attributes like modifiability, security, and availability, the new method gives importance to other quality attributes that are significant to the ecosystem vision such as affordability, scalability, and interoperability.
- *Stakeholders*: In addition to the internal stakeholders and the evaluation team, the envisioned method integrates external SECO stakeholders, because anticipating needs and goals of external partners is vital for SECO success.
- *Output*: The envisioned method will extend reports on sensitivity points, tradeoff points, risks, and non-risks to include interactions with external stakeholders and their relative systems.
- *Process*: SECOs are dynamic systems, formed by the continuous interactions of its participants. It is crucial that the envisioned method provides mechanisms for continuous and agile architecture evaluation.

II. METHODOLOGY

Design of the EAM is based on empirical research methods. We started with a series of workshops with representatives of two large Swedish companies. By this, we contextualized the research problem to the companies' ecosystems and identified a set of themes with potential impact on the design of the technological platform along which software ecosystems can be analyzed. Example themes include nature of the ecosystem, governance models, and software platform (if there is one).

Our next step was to conduct a combination of literature review and archival study to refine the set of themes and identify the relevant discussion subtopics for each theme. As the perspective of this work is software ecosystems, we have started with reviewing two recent summative works on SECOs: a systematic literature review [3] and an edited book on the state-of-the-art of software ecosystems [4].

Having identified the subtopics of interest, our next step was to conduct a detailed archival study of example software ecosystems (e.g. Eclipse, Android, SAP). The goal was to illustrate the topics of interests identified in the form of a comparative study of the example ecosystemability.

III. ECOSYSTEMABILITY AND OTHER QUALITY ATTRIBUTES

Similar to existing assessment methods, EAM is quality-centric. It is thus important to depict where ecosystemability stands with respect to other quality attributes and quality models such as the widely used ISO/IEC standard [5].

In their systematic literature review on software ecosystems [3], Manikas et al. present a set of qualities for SECO architecture and identified the following quality characteristics: *Understandability*. SECO actors should be able to easily comprehend the ecosystem and its underlying technological platform, e.g. the nature of the ecosystem, its business rules and restrictions, as well as extension and configuration options. *Expandability*. A good SECO architecture should allow different ecosystem actors to accommodate additions to the core capabilities. *Expandability* also means to be able to customize and configure existing platform features.

Parallel development and testability. SECO actors typically develop their platform additions independently. A good technological platform should enable parallel independent development. In particular it should support testing of platform additions in the context of the SECO actors.

Maintainability. One of the major requirements for SECO architecture is to support ecosystem actors through improving and evolving the original technical platform to be able to create new business value.

Deployment and time to market. On the business side, a SECO architecture should allow for fast and seamless deployment of different versions of the product in different usage contexts. Easier deployment means faster time to market, which is a critical business factor for an ecosystem's industrial actors.

We believe that the above list of quality properties is a good starting point for a quality model for ecosystemability. Through collaboration with industrial partners we intend to construct a refined model and assessment framework.

IV. EXAMPLE ECOSYSTEMABILITY ASSESSMENT THEMES

Table 1 gives examples of our EAM assessment themes. The first column relates to the theme, the second gives examples of assessment criteria, and for some (those with white background) we give a short explanation on how this can affect architectural decisions about the technological platform. Together, these themes form an elicitation and analysis guide that allows characterizing a SECO under investigation as well as capturing business requirements and contextual properties that impact decisions about the evolution of the technological platform. We successfully applied this guide in a workshop setting (60-90 min.), which allowed us to compare the SECO under investigation with other SECOs that we previously assessed (e.g. in our archival study, see Sect. II) and to give

TABLE I
EXAMPLES OF EAM ASSESSMENT THEMES

| Theme | Item | Significance |
|---------------------|--|---|
| Nature of ecosystem | <ul style="list-style-type: none"> – Degree of openness – Level of ecosystem stack – Extension market | There is a tradeoff between the need to protect intellectual property and the need to be open with partners in the SECO. Certain architectural styles (e.g. core periphery) promise mitigation. |
| Governance | <ul style="list-style-type: none"> – Roadmapping – Niche creation – Entry barriers | A SECO with a high entry barrier (e.g. SAP) can afford lower understandability than a SECO with a low entry barrier (e.g. Android), aiming for not discouraging a continuous stream of new actors. |
| Software platform | <ul style="list-style-type: none"> – Extension model – Deployment activities supported – Operation supported by extensions | Reasons to limit the freedom and power of extensions include the ability for non-functional testing of the whole system or to protect the business model of the SECOs market place. The architecture of the technological platform could be designed to enforce such decisions. |

examples on how others dealt with similar challenges. In the future EAM will allow comparing the ecosystemability of different architectural alternatives.

V. CONCLUSION AND OUTLOOK

In this paper we present our research plan and preliminary results towards the development of the ecosystemability assessment method (EAM). EAM will offer platform providers in SECOs a checklist for significant aspects of ecosystemability and their assessment, best practices and guidelines for evolving the underlying technological platform, and a way to uncover opportunities in the ecosystem and to articulate its goals. Based on our preliminary results, we strongly believe that this will strengthen communication between ecosystem actors and help to make informed technical decisions in software ecosystems.

ACKNOWLEDGEMENT

This research is funded by the Software Center and we thank the companies and their representatives who participated in our work so far for their enthusiasm and support.

REFERENCES

- [1] S. Jansen and G. Capelleveen, *Quality review and approval methods for extensions in software ecosystems*. Edward Elgar Publishing, 2013, ch. Software Ecosystems, pp. 187–217.
- [2] “ATAM: Method for Architecture Evaluation.” last accessed April 2014. [Online]. Available: <http://www.sei.cmu.edu/library/abstracts/reports/00tr004.cfm>
- [3] K. Manikas and K. M. Hansen, “Software ecosystems - a systematic literature review,” *Journal of Systems and Software*, vol. 86, no. 5, pp. 1294–1306, 2013.
- [4] S. Jansen, M. Cusumano, and S. Brinkkemper, Eds., *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Edward Elgar Publishing, 2013.
- [5] “ISO/IEC 25010:2011,” last accessed April 2014. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733

Combined Goal and Feature Model Reasoning with the User Requirements Notation and jUCMNav

Yanji Liu, Yukun Su, Xinshang Yin, Gunter Mussbacher

ECE, McGill University
Montréal, Canada

{yanji.liu | yu.k.su | xinshang.yin} @mail.mcgill.ca, gunter.mussbacher@mcgill.ca

Abstract— The User Requirements Notation (URN) is an international requirements engineering standard published by the International Telecommunication Union. URN supports goal-oriented and scenario-based modeling and analysis. jUCMNav is an open-source, Eclipse-based modeling tool for URN. This tool demonstration focuses on recent extensions to jUCMNav that have incorporated feature models into a URN-based modeling and reasoning framework. Feature modeling is a well-establishing technique for capturing commonalities and variabilities of Software Product Lines. Combined with URN, it is possible to reason about the impact of feature configurations on stakeholder goals and system qualities, thus helping to identify the most appropriate features for a stakeholder. Furthermore, coordinated feature and goal model reasoning is fundamental to Concern-Driven Development, where concerns are defined with a three-part variation, customization, and usage interface. As the variation interface is described with feature and goal models, it is now possible with jUCMNav to define and reason about a concern’s variation interface, which is a prerequisite for composing multiple concerns based on their three-part interfaces.

Index Terms—goal modeling, feature modeling, GRL, Goal-oriented Requirement Language, URN, User Requirements Notation, trade-off analysis, evaluation algorithm, concern, concern-driven development, jUCMNav.

I. INTRODUCTION

The User Requirements Notation (URN) is an international requirements engineering (RE) standard published by the International Telecommunication Union in the Z.15x series [5]. URN targets the modeling and analysis of goals and scenarios, by combining the Goal-oriented Requirement Language (GRL) with the Use Case Map (UCM) notation. For more than a decade [2], URN has proven and continues to be useful and successful for many RE activities, in academia and industry. The free, open-source, Eclipse-based jUCMNav tool has contributed significantly to this success, as it allows URN models to be defined, navigated, analyzed, and transformed [6].

Recently, it was suggested to combine goal-oriented and feature-based modeling for more sophisticated reasoning [7] in the context of Software Product Lines (SPLs) [8] and Concern-Driven Development (CDD) [1]. SPLs are a structured reuse approach and have traditionally used feature models to identify the commonalities and variabilities of a family of similar

products. The objective of Concern-Driven Development (CDD) is to provide next generation reuse technologies, through a three-part interface describing units of reuse, i.e., concerns. The variation interface describes required design decisions and their impact on high level system qualities with the help of feature and goal models. The customization interface allows the chosen variation to be adapted to a specific reuse context, while the usage interface defines how a customized concern may eventually be used.

The next section introduces three contributions to facilitate combined reasoning of feature and goal models in jUCMNav: (i) support for the definition of feature models based on goal model semantics, (ii) simultaneous analysis of feature and goal models through a single, shared evaluation algorithm, and (iii) improved analysis based on feature model semantics capable of identifying invalid feature configurations and automatically selecting mandatory features.

II. FEATURE MODELS AND URN

To support the modeling and analysis of feature models in jUCMNav, the metamodel for GRL was extended with feature modeling concepts by subclassing existing GRL metaclasses. A *feature* (\square , e.g., Energy Management in Fig. 1) is a subclass of GRL intentional elements (i.e., the nodes in a GRL model). A \square is chosen as features are conceptually close to GRL tasks (also shown with \square), while a \square represents the very different concept of GRL resources. *Optional links* (—○, e.g., between Energy Management and Air Conditioning Control) and *mandatory links* (—●, e.g., between Energy Management and Light Control) are subclasses of GRL contribution links. Since feature models are an and/or tree and goal models are an and/or graph, *alternative feature groups* and *OR feature groups* can be modeled with existing XOR and OR decomposition links in GRL, respectively (note that AND decomposition links are an alternative way of modeling mandatory links). Cross-tree integrity constraints such as “requires” or “conflicts” can be defined with OCL constraints [7], which have been supported by jUCMNav since version 3.1. Finally, a *feature configuration* is a subclass of the GRL strategy concept. Both allow a set of model elements to be selected for analysis purposes.

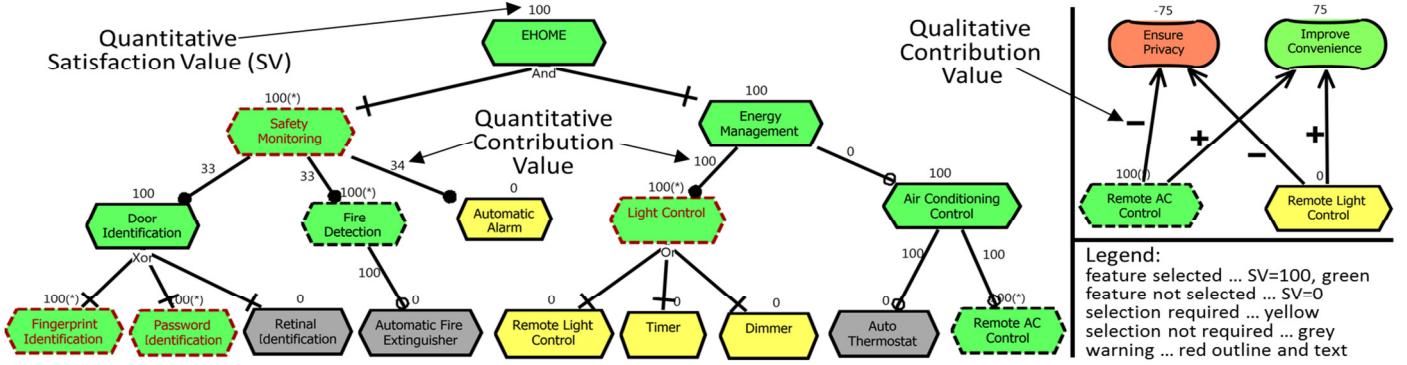


Fig. 1. Feature Modeling and Analysis in jUCMNav

In GRL, each selection can be quantified with a satisfaction value between -100 (element is not satisfied at all) and 100 (fully satisfied), while in the context of feature models, the selection is constrained to 0 (for not part of the feature configuration) and 100 (for part of the feature configuration).

The advantage of this approach is that (i) the infrastructure for evaluation algorithms [3] is readily available for feature models and (ii) features can be used in GRL models to define the impact of features on stakeholder goals and system qualities, thus enabling combined reasoning about feature and goal models for stakeholder trade-offs. The single evaluation algorithm shared between feature and goal models only needs to understand how to interpret optional and mandatory links, because features/intentional elements, decomposition links/feature groups, feature configuration/strategies, and integrity/OCL constraints are semantically equivalent in terms of their required behavior for the evaluation algorithm. This new evaluation algorithm is called the “Feature Model Strategy Algorithm” and can be selected in the preference settings of jUCMNav. The result of a feature model evaluation with the customary color-coded feedback is shown on the left side of Fig. 1. Note that standard GRL goal models capture the impact of features on goals (e.g., Remote Light Control and Remote AC Control have a negative impact on Ensure Privacy but a positive impact on Improve Convenience) as shown briefly on the right.

In addition, several warnings have been implemented that indicate with a red dashed outline/text to the modeler where in the feature model a feature configuration is currently invalid (in Fig. 1, e.g., a mandatory child of Safety Monitoring is not selected, at least one child of Light Control is not selected, and at the most one child of Door Identification can be selected). Optionally, auto-selection of mandatory features can be activated via jUCMNav’s preferences. If activated, all mandatory and AND-decomposed children of a feature are automatically selected, when evaluating the feature model.

III. CONCLUSIONS AND FUTURE WORK

This paper gives an overview of three recent extensions to jUCMNav, the currently most comprehensive URN modeling tool, for the purpose of integrated feature and goal model reasoning. This enables jUCMNav to be used in the context of SPL development, in general, and for CDD in particular. In the context of goal modeling for SPLs, several

approaches [4][9][10] offer only some but not all of (i) feature modeling, (ii) the modeling of impact of features on goals, (iii) reasoning about stakeholder trade-offs, and (iv) tool support in contrast to URN/jUCMNav. In the context of CDD, jUCMNav is currently the only tool that supports requirements engineering activities for CDD. In future work, we plan to formalize the composition of concerns based on the three-part interface and support such composition in the jUCMNav tool.

REFERENCES

- [1] Alam, O., Kienzle, J., and Mussbacher, G., “Concern-Oriented Software Design”, Model Driven Engineering Languages and Systems, Springer, 2013, LNCS 8107:604–621. DOI: 10.1007/978-3-642-41533-3_37.
- [2] Amyot, D. and Mussbacher, G., “User Requirements Notation: The First Ten Years, The Next Ten Years”, Journal of Software (JSW), 2011, 6(5):747–768. DOI: 10.4304/jsw.6.5.747-768.
- [3] Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., and Yu, E., “Evaluating Goal Models within the Goal-oriented Requirement Language”, IJIS, Wiley, 2010, 25(8):841–877. DOI: 10.1002/int.20433.
- [4] Bagheri, E., Noia, T., Ragone, A., and Gasevic, D., “Configuring Software Product Line Feature Models Based on Stakeholders’ Soft and Hard Requirements”, Software Product Lines: Going Beyond, Springer, 2010, LNCS 6287:16–31. DOI: 10.1007/978-3-642-15579-6_2.
- [5] International Telecommunication Union, “Recommendation Z.151 (10/12), User Requirements Notation (URN) – Language definition”, 2012. <http://www.itu.int/rec/T-REC-Z.151/en>
- [6] jUCMNav website, version 6.0, University of Ottawa. <http://jucmnav.softwareengineering.ca/jucmnav>
- [7] Mussbacher, G., Araújo, J., Moreira, A., and Amyot, D., “AoURN-based Modeling and Analysis of Software Product Lines”, Software Quality Journal (SQJ), Springer, 2011, 20(3-4):645–687. DOI: 10.1007/s11219-011-9153-8.
- [8] Pohl, K., Böckle, G., and van der Linden, F.J., “Software Product Line Engineering: Foundations, Principles and Techniques”, Springer, 2005.
- [9] Than Tun, T., Boucher, Q., Classen, A., Hubaux, A., and Heymans, P., “Relating Requirements and Feature Configurations: A Systematic Approach”, 13th Intl. Software Product Line Conference (SPLC’09), CMU, 2009, 201–210.
- [10] Yu, Y., Leite, J., Lapouchian, A., and Mylopoulos, J., “Configuring Features with Stakeholder Goals”, SAC’08, ACM, 2008, 645–649. DOI: 10.1145/1363686.1363840.

Decisively: Application of Quantitative Analysis and Decision Science in Agile Requirements Engineering

Sanjaya Kumar Saxena
GrayPE Systems (P) Limited
NOIDA, India
sanjaya@graype.in

Rachna Chakraborty
GrayPE Systems (P) Limited
NOIDA, India
rachna@graype.in

Abstract—While many mature Requirements Engineering (RE) tools for Agile exist, RE professionals at large have not been able to benefit from Quantitative Analysis and Decision Science (QUADS) techniques in this context. In this paper we present an Agile RE tool, *Decisively*, which brings a new perspective to automation in the RE process through application of QUADS to address Requirement Discovery, Analysis, Estimation and Prioritization. Techniques explored in *Decisively* include Analytical Hierarchical Process (AHP) for prioritization and estimation, Lorenz function to shortlist user stories by analyzing the distribution of votes, Box Plot Analysis to predict velocity, and Text Mining to discover implied requirements from documents.

Index Terms—Agile, Quantitative Analysis & Decision Science, Requirements Prioritization, Story Points Estimation, Velocity Prediction, Box Plot, Text Mining, AHP, SPAN.

I. INTRODUCTION

There are many mature industrial-strength RE tools for Agile, but the application of Quantitative Analysis and Decision Science (QUADS) in these tools has been largely limited to analytics and creation of data visualizations. The direct use of specific QUADS tools in the process of RE has been minimal due to the inaccessibility of affordable tools and the complexities involved in their usage. For instance, (a) in order to text-mine implied requirements from a policy document [1], a tool like R is required along with the knowledge of mining techniques, and R-scripting; (b) to prioritize using Analytical Hierarchical Process (AHP), specialized tools are needed and it presents challenges in the process of achieving consistency together with consensus in a group setting [2, 3]; and (c) even the simple task of short listing multi-voted items needs analysis of distribution of votes.

Decisively is an initiative to remove these barriers in large-scale adoption of QUADS techniques, and is an extension of earlier research [4] to aid non-decision science professionals in decision-making. It is a web-based tool with a special focus on usability, which is achieved by masking the complexities of QUADS techniques from end-users. It deploys (a) Text Mining to extract implied requirements from documents, and to detect similar user stories during ideation; (b) Lorenz curve to shortlist user stories by analyzing the distribution of votes; (c) AHP for prioritization and estimation, along with fast algorithms to provide real time feedback on inconsistent judgments; (d) Clustering to identify different schools of

thoughts during prioritization and estimation; (e) Successive Proportional Additive Numeration (SPAN) to arrive at consensus [5]; and (f) Box Plot Analysis to detect outliers, and predict velocity with confidence interval from past iterations.

Prior to initiating the research and development on the current version of *Decisively*, these techniques were presented and discussed with practitioners in a SCRUM Alliance approved conference where it was received very well [6]. Application of *Decisively* in Agile projects in context of RE is outlined in the next section.

II. APPLICATION OF DECISIVELY

A. Identification of user stories

User stories are typically created in brainstorming sessions involving the stakeholders. The focus of such sessions is to eventually build a complete and mutually exclusive set of stories. These stories must be mutually exclusive to achieve flexibility in prioritization & sprint planning. *Decisively* provides an intelligent, near real-time ideation mechanism to make this possible. It aids each participant as s/he enters a new story by detecting and displaying similar stories in real time (Figure 1). This is extremely useful when participants are collaborating remotely or in several small sessions instead of a common one. In order to review the completeness and mutual exclusivity of stories, *Decisively* extracts roles, and clusters similar goals and similar reasons by mining the user stories.

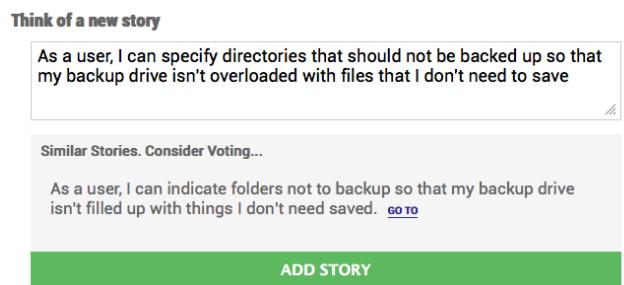


Fig. 1. Detection of similar story using text mining during ideation

In addition, *Decisively* provides a mechanism to rapidly sift through large documents using text mining. Discovering details for stories from policy, rule and/or compliance documents is a very tedious, time consuming and error prone tasks. To counter this, apart from highlighting important content,

Decisively provides smart navigation mechanisms within the document for discovery of information. By identifying frequently used text patterns, *Decisively* allows users to navigate sentences and paragraphs against a pattern they select. This reduces the time and tedium needed to uncover critical requirements, thereby reducing the probability of missing them.

B. Prioritization

In most scenarios, Prioritization happens based on subjective judgments, except in few cases where a quantifiable economic case may be available. Subjective judgments inherently suffer from varying degrees of inconsistency arising out of human thinking. It is further complicated when multiple stakeholders are involved. They can belong to different schools of thought, and this makes discussing differences meaningfully and arriving at a group priority, difficult.

Decisively leverages AHP for prioritization. Unique pairs of requirements are successively presented to stakeholders and they have to indicate their degree of preference towards one story on a scale of 1 to 9. While making a choice, if there is any inconsistency with the previous judgments they have made, it is highlighted in real time (Figure 2). *Decisively* aggregates judgments from all stakeholders to form a Group Priority. During the aggregation process, clusters of similar rankings from different stakeholders are automatically identified to display different schools of thought, if there are any. Any group level inconsistencies arising out of differences in viewpoints are also highlighted, along with recommendations on how they can be resolved. In extreme situations where differences cannot be resolved even after discussions, *Decisively* offers a mechanism of arriving at a general agreement using techniques like SPAN to reach a consensus on priority rankings.

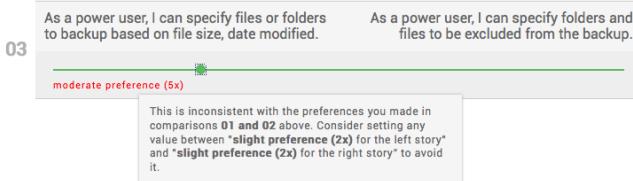


Fig. 2. Real time feedback of inconsistency during AHP based prioritization

C. Estimation

Similarly, *Decisively* utilizes AHP for estimation of story points. All stories that need to be estimated are presented to the user with one or two stories of known size for pair-wise comparison. Using the relative weights determined by the pair-wise comparison, and the story points of the known user stories, *Decisively* is able to compute their story points. The system speeds up such discussions considerably because it clearly explains the difference or alignment in views (Figure 3). It also allows effective estimation of more than one user story in a single comparison session.

Further, *Decisively* is very useful in determining the number of stories that can be completed in an upcoming iteration of development. It analyzes the variation in velocities

of past iterations, detects any outliers, and then predicts the number of story points that can be covered with a confidence interval.

Group Alignment

The group is **reasonably aligned** on estimates, as seen from the average deviation. The infographics below highlights the individual weight difference from the group in the increasing order of difference.

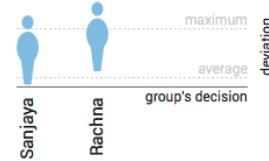


Fig. 3. Alignment of group's view during AHP based estimation

III. CONCLUSION & FUTURE WORK

In creating *Decisively*, QUADS techniques have been optimally leveraged to address real-world Agile RE challenges. Initial user testing confirms that users were able to make the best use of *Decisively* even without any prior knowledge of QUADS techniques. Users also appreciated the unique simplicity of *Decisively*, the meaningful inferences drawn by the tool, and acknowledged its potential in improving their productivity. Work is in progress for releasing a beta version. Its future roadmap includes addressing the scalability of prioritization using AHP, addition of TOPSIS and KANO model as alternatives, and development of an API for integration with other tools.

REFERENCES

- [1] Aaron K. Massey et al., "Automated text mining for requirements analysis of policy documents," 21st IEEE Int. Requirements Engineering Conf., pp. 4–13, July 2013.
- [2] Perini, A. et al., "An Empirical Study to Compare the Accuracy of AHP and CBRanking Techniques for Requirements Prioritization", 5th Int. Workshop on Comparative Evaluation in Requirements Engineering, pp. 23–35, 2007.
- [3] Witold Pedrycz and Mingli Song, "Analytic Hierarchy Process in Group Decision Making and its Optimization With an Allocation of Information Granularity," IEEE Transactions On Fuzzy Systems, Vol. 19, No. 3, pp. 527–539, June 2011.
- [4] Sanjaya Kumar Saxena and Rachna Chakraborty, "Aiding non decision-science professionals in making a decisive move," presented at 24th Subjective Probability, Utility, and Decision Making (SPUDM) Conf., Barcelona, Spain, 2013. Available <http://decisive.ly/pdf/decisively.pdf>
- [5] William J. Mackinnon, "Elements of the SPAN Technique for Making Group Decisions," The Journal of Social Psychology, 70:1, pp. 149–164.
- [6] Sanjaya Kumar Saxena (2013, Nov. 16) "Statistics & Decision Science for Agile," [Online]. Available <http://agilenoida.files.wordpress.com/2013/10/stats-ds-for-agile.pdf>

VARED: Verification and Analysis of Requirements and Early Designs

Julia Badger

NASA- Johnson Space Center, USA

julia.m.badger at nasa.gov

David Throop

The Boeing Company, USA

david.r.throop at boeing.com

Charles Claunch

Oceaneering Space Systems, USA

charles.a.claunch at nasa.gov

Abstract—Requirements are a part of every project life cycle; everything going forward in a project depends on them. The VARED tool chain aims to provide an integrated environment to analyze and verify the requirements and early design of a system. Natural language requirements are processed automatically into formal specifications using a state model of the system under design and its environment. The specifications are formally checked and then are used to verify the controller model meets the requirements.

I. MOTIVATION

Requirements are a part of every project life cycle; everything going forward in a project depends on them. Good requirements are hard to write, there are few useful tools to test, verify, or check them, and it is difficult to properly marry them to the subsequent design, especially if the requirements are written in natural language. In fact, the inconsistencies and errors in the requirements along with the difficulty in finding these errors contribute greatly to the cost of the testing and verification stage of flight software projects [1].

Large projects tend to have several thousand requirements written at various levels by different groups of people. The design process is distributed and a lack of widely accepted standards for requirements often results in a product that varies widely in style and quality. A simple way to improve this would be to standardize the design process using a set of tools and widely accepted requirements design constraints. The difficulty with this approach is finding the appropriate constraints and tools. Common complaints against the tools available include ease of use, functionality, and available features. Also, although preferable, it is rare that these tools are capable of testing the quality of the requirements.

II. TOOL DESCRIPTION

The VARED tool chain (Figure 1) aims to provide an integrated environment to analyze and verify the requirements and early design of a system. The input to the tool are natural language requirements that have been written to some common standards. The requirements are then automatically parsed into a formalized requirements language and checked for quality using Natural Language Processing (NLP) techniques. The Requirements Conversion Engine (RCE) then automatically converts the formalized requirements into Linear Temporal Logic (LTL) statements. LTL is an appropriate form for using formal methods both on and with the requirements. The

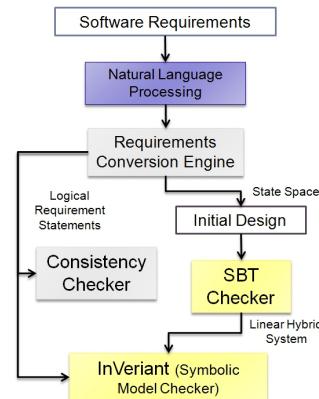


Fig. 1: VARED- integrated requirements and early design tool chain.

Logical Consistency Checker (LCC) will formally analyze the LTL version of the requirements for satisfiability and vacuity with respect to an appropriate state model, seamlessly incorporating automatic testing and verification of the requirements statements into the design process.

Two existing tools have been adapted to design and verify early system designs based on environment models and the formal requirements statements. The SBT Checker and the state model will help designers create controllers and system models with the appropriate structure needed for the InVeriant symbolic model checker. InVeriant will formally verify the controllers or system model against the LTL version of the requirements statements, which both incorporates automatic testing and verification for the early design stage of software development, and also formally ties the requirements and early design together.

A novel feature of the VARED toolchain is the use of a state model of the system and its environment. This state model is developed from the scope and the high-level requirements of the system under design. The model is created using State Analysis-derived methods [2] and is used to aid nearly all pieces of the VARED tool chain, including the natural language processing, the LCC, SBT Checker, and InVeriant. Brief descriptions of the tool's components are as follows.

The natural language processing tool for VARED is called Edith and was built on more than a decade of work on the underlying STAT parsing tool. STAT (Semantic Text Analysis Tool) [3] is a NLP tool that was originally built to parse

through space shuttle problem reports. The Edith extension allows STAT to be used to parse requirement statements into a structure that can be parsed by the Requirements Conversion Engine, if possible. Edith parses the sentences structure, which is naturally normalized by the fairly structured English language found in requirements statements, and replaces common words and phrases with general words associated with LTL semantics. It also queries the state model to find the appropriate form that the subject and other noun phrases should take, so that the formal requirement statements can be used with the state model downstream in the tool chain. If Edith encounters a sentence structure or noun phrase that it does not recognize, the requirement statement is returned to the user with an error message; otherwise, a structured English statement conforming with the state model terminology is sent to the RCE.

The RCE consists of two parts. The first part is a third-party open source package called SALT that translates structured English to LTL. The second part takes the LTL output and converts it to a form that can be used by the downstream tools. SALT (Structured Assertion Language for Temporal Logic) [4] is a temporal specification language designed to create concise statements that are used in model checking and runtime verification. In particular, SALT is configured to output LTL as understood by SMV (Symbolic Model Verifier) [5]. The RCE returns requirements statements written in LTL in PANDA syntax [6], which is used by the Logical Consistency Checker and InVeriant.

The LCC provides two functionalities for checking the quality of the formal requirement statements against the state model. First, it uses PANDA [6] to format the LTL formulas in conjunction with formalized state model properties to create an SMV input file. This input file checks the given formula against a universal model to check the satisfiability of the statement with respect to the state model. Second, the LCC will take the combined requirements and state model LTL statements and check them for vacuity [7]. Informally, an LTL formula is vacuous in an expression if that expression does not affect the value of the formula. The LCC provides the user feedback on requirements statements that are vacuous or unsatisfiable so that the user can correct the initial requirement or the state model of the system as necessary.

InVeriant is a symbolic model checker that is capable of doing safety checking on a class of linear hybrid automata (LHA) that have a special property called state-based transitions [8]. This property, which can be checked using the SBT Checker, ensures that the system covers the state model- that is, each state represented in a model of the system and its environment has a corresponding location in the model of the controller. The InVeriant model checker then combines the individual LHA into the model for the system under control using both the controller LHA inputs and the state model of the system. It then checks the overall system design against the formal requirements statements to ensure that they hold in all states of the system design. Both tools provide the user feedback if the models do not pass the respective checks, allowing the user to redesign the system model as appropriate to meet the

state-based transitions property or to satisfy the requirement statement.

III. CONTRIBUTIONS AND FUTURE WORK

Though natural language processing of requirements is broadly studied (i.e., [9], [10]) and the formal methods algorithms utilized in the VARED tool are on par with current technology, the major contribution in this work is combining the two while taking a system-level approach. The design method that this tool uses requires the requirements engineer to create a model of the system and the environment from the requirements and supporting documentation. Then, using that model, the VARED tool helps designers to both analyze the quality of requirements as well as provides traceability of the requirements through the early design stage, where the controller is designed and verified against the requirements. This state model based approach is unique to both natural language processing and symbolic model checking tools available today.

The VARED tool has been developed to support systems whose controllers can be described as linear hybrid automata with certain properties. Because the properties and controller design depend heavily on the fidelity of the state model constructed, this tool is appropriate for analyzing and developing systems in the early stage of design. The choice of LTL as the specification language introduces limitations on the set of requirements that can be analyzed; however, the tool is expandable to other types of logics as needed. Several other future directions for the model checkers could be implemented to expand the usefulness of the VARED tool, such as introducing timing, liveness properties, and branching.

REFERENCES

- [1] D. Peercy, *Software Quality Engineering Course Guide*. SEMATECH, 1995.
- [2] D. Dvorak, M. Indictor, M. Ingham, R. Rasmussen, and M. Stringfellow, “A unifying framework for systems modeling, control systems design, and system operation,” *IEEE Conference on Systems, Man, and Cybernetics*, October 2005.
- [3] J. T. Malin, C. Millward, F. Gomez, and D. R. Throop, “Semantic annotation of aerospace problem reports to support text mining,” *IEEE Intelligent Systems*, vol. 25, pp. 20–26, 2010.
- [4] A. Bauer, M. Leucker, and J. Streit, *SALT: Structured Assertion Language for Temporal Logic*. Springer Berlin / Heidelberg, 2006, vol. LNCS 4260, pp. 757–775.
- [5] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang, “Symbolic model checking: 10^{20} states and beyond,” in *Proc. of the Fifth Annual IEEE Symposium on Logic in Computer Science*, 1990, pp. 428–439.
- [6] K. Y. Rozier and M. Y. Vardi, “A multi-encoding approach for LTL symbolic satisfiability checking,” in *FM 2011: Formal Methods*. Springer, 2011, pp. 417–431.
- [7] A. Gurfinkel and M. Chechik, “Robust vacuity for branching temporal logic,” *ACM Transactions on Computational Logic (TOCL)*, vol. 13, no. 1, p. 1, 2012.
- [8] J. M. B. Braman, “Safety verification and failure analysis of goal-based hybrid control systems,” Ph.D. dissertation, California Institute of Technology, 2009.
- [9] L. Mich and R. Gariglano, “NL-OOPS: A requirements analysis tool based on natural language processing,” in *Proceedings of Third International Conference on Data Mining Methods and Databases for Engineering, Bologna, Italy*, 2002.
- [10] F. Iwama, T. Nakamura, and H. Takeuchi, “Constructing parser for industrial software specifications containing formal and natural language description,” in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE ’12, 2012, pp. 1012–1021.

Structured Multi-view Modeling by Tabular Notation

Xiuna Zhu

Institut für Informatik

Technische Universität München

Boltzmannstr. 3, D-85748, Germany

Email: zhux@in.tum.de

Dongyue Mou

Fortiss GmbH, Guerickestr. 25,

D-81735 München, Germany

Email: mou@fortiss.org

Daniel Ratiu

Fortiss GmbH, Guerickestr. 25,

D-81735 München, Germany

Email: ratiu@fortiss.org

Abstract—The growth of software complexity and high degree of dependencies between functionalities motivates the use of models during requirements engineering. Hence, readability and comprehensibility of currently requirements specification techniques should be increased. Additionally, multi-view modeling and tabular expression are widely accepted techniques in requirements documentation. We present a tool that allows structured multi-view modeling of the behavior of the system by means of tabular notation. Our tool provides various table patterns to support different behavior views, which leverage the advantages of tabular specification, e.g., unambiguous, precise, and easier to read, analyses and communicate. Our aim is to reduce the complexity in the development of software systems.

Index Terms—Tabular Specification, Multi-View Modeling, Model-based Requirements Engineering

I. INTRODUCTION

Nowadays software systems are employed in every domain. However, the growth of their complexity and high degree of dependencies between functionalities demands adequate specification approaches. Variety of approaches, such as model-driven requirements engineering, have been proposed to face the difficulties in the development of software systems. Multi-view modeling is considered among them as a technique to reduce the complexity [1].

Tabular expressions are used in the process of requirements documentation by clarifying the logical formulas [2]. Therefore, software engineers often use tables to specify the behavior of components in a system, but only as documentation. We advocate that tables can be used beyond the documentation. Different kinds of tables can be used to specify system interfaces and behaviors. And we can analyse their functional properties by analyse the tables. In so doing, tabular specification is no more only considered as documentation but exploited as a model of the system.

Due to tabular description techniques' advantages, e.g., they are unambiguous, precise, and intuitive, they are promising for practical system development [3]. Several types of tables have been defined in previous researches of tabular notation for state machine based models [4], [5]. This paper focuses on tabular specification, namely we take tables as pragmatic specification formalism of systems' behavior. For one thing, by the definition of the tabular notation for modeling and translating from tables into formulas of predicate logic and vice

versa, we provide a bridge between the conciseness of readable specification and the preciseness of mathematical methods. For another, by manipulation of the tabular specification, we attempt to obtain the composition of the tabular specification.

To achieve the tool support for tabular specification, we have focused our study on following main research questions.

Q1) How to specify the functional requirements by tabular specification? This research question is concerned with how to use tables as a pragmatic specification formalism for a both precise and readable specification of system's functional requirements. Providing multiple consistent views of the system on different levels of abstraction was discussed and the definition of the tabular notation was provided.

Q2) How to clarify formal semantics of tabular specification? The formal system model should be used to define the semantics of the employed description techniques.

Q3) How to cross the functional boundary of single component to obtain a systematical view of the functionalities? To achieve the composition and decomposition of the tabular specification, we discuss on how to compose tabular-specified models and ensure the consistency and completeness of the composite model.

Q4) How to derive the optimization of system architecture from the tabular specification? To analysis the property of the system which specified by tabular specification, we can via the analysis of the property of tables. By translating tables into logical formulas, which define precise semantics for them, the dependency relation between outputs and inputs will be achieved. By dependencies encapsulated and automatically resolved, the structure features and refinement of system can be obtained, and the optimization of system architecture by manipulation on tabular specification can be achieved.

II. VARIOUS TABULAR PATTERNS FOR THREE VIEWS

The proposed tabular modeling tool is the implementation and extension of the previous work [6]. One goal is that our tool supports the modeling process in a user-friendly manner. Various table patterns with manifold structures are provided. Therefore, user can choose the adaptive format as table pattern or customize the structure of the tables.

In our case, a system can be structured into a set of logical entities, which are connected via communication channels.

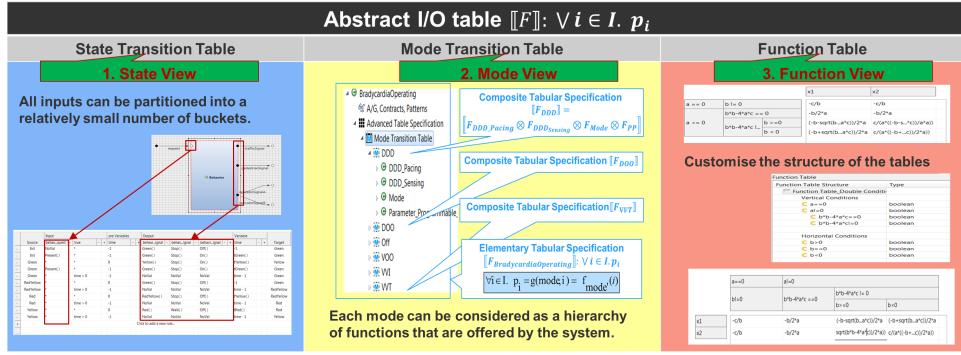


Figure 1. Supported Table Patterns for Tabular Specification

Each table can specify a functionality of system, which is a combination in form of a predicate of logical entities. The structured tabular models are classified to two types: *Elementary Tabular Specification* and *Composite Tabular Specification*. Tabular specification is suitable for describing the structured composition and decomposition of such systems by the manipulation on tables which are used to describe the system.

As shown in Fig.1, three modeling views are supported for elementary tabular specification. To achieve those views, we provided three basic table patterns: state transition tables, mode transition tables, and function tables. Here, modes represent the logical states of a functionality. Each mode can be considered as a hierarchy of functionalities that are offered by the system. Each table pattern has a general specific table structure, but the structure of the table can be customized arbitrary. So the plasticity of table structure can be kept in the proposed tabular specification.

III. ILLUSTRATIVE CASE STUDY

Unlike requirement documents written in natural language, which allow limited semantic and syntactic checks, we captured and modeled the requirements precisely by proposed tabular specification tool. It is integrated in model-based development tool AutoFOCUS3. Many case studies are carried out with tabular specification in AF3 modelling environment, and tool demos are available at [7]. We illustrate the functionality of our tool by the use of state transition tables for a traffic lights control system, mode transition tables for the pacemaker challenge project, and function tables for a quadratic equations solver.

Except specifying the requirements in a well-structural format, the tool concentrates on the questions if the tabular models match the customer's requirements. The problems which is not easy to find in the requirements document, like if the requirement is complete and consistent, if there is a conflict existing between two requirements, can be detected by tool.

IV. CONCLUSION

This work presents the following contributions to the current researches in formal specification:

1) From tabular expression to tabular specification

Based on the FOCUS modeling theory, we define the notion of tabular specification. Instead of using graphical notation to describe the behavior of interactive systems, we define the tabular notation for state machine-based specifications to obtain a tabular specification of system behavior. By manipulation of the tables, such as composition and projection, we yield tabular specified components and the composition of the components.

2) Tool support for various table patterns The tabular specification tool supports tabular specification, simulation, property analysis, and code generation. Various table patterns with manifold structures are provided and supported, which can help shortening the cells content and bring tables to a nice form.

3) Structuring features for complex systems The large number of states is a defect which affects application of current specification techniques to describe the behavior of practical systems. Structured tabular specification was proposed to describe the behavior of the hierarchically tables. The illustrative case studies showed that in structural multi-view tabular specification the number of states can be significantly reduced and the complexity of behavior can be reduced.

REFERENCES

- [1] M. Broy, "Software and system modeling: Structured multi-view modeling, specification, design and implementation," in *Conquering Complexity*, M. Hinckley and L. Coyle, Eds. Springer, 2012, pp. 309–372.
- [2] R. Janicki and A. Wassyng, "Tabular expressions and their relational semantics," *FUNDAMENTA INFORMATICAE*, vol. 67, no. 4, pp. 343–370, 2005.
- [3] M. K. Zimmerman, K. Lundqvist, and N. G. Leveson, "Investigating the readability of state-based formal requirements specification languages," in *ICSE*, W. Tracz, M. Young, and J. Magee, Eds. ACM, 2002, pp. 33–43.
- [4] I. Bourguiba and R. Janicki, "Table-based specification techniques," in *Computers Industrial Engineering, 2009. CIE 2009. International Conference on*, July 2009, pp. 1520–1525.
- [5] M. Herrmannsdörfer, S. Konrad, and B. Berenbach, "Tabular notations for state machine-based specifications," *Crosstalk*, vol. 21, no. 3, pp. 18–23, 2008.
- [6] X. Zhu, "A formal model for service-based behavior specification using stream-based i/o tables," in *FACS2013*, Nanchang, China, 2013.
- [7] Fortiss. (2013, Sep.) Homepage of the autofocus 3. [Online]. Available: <http://af3.fortiss.org>

Efficient Visual Notations for Efficient Stakeholder Communication

Ralf Laue

University of Applied Sciences
of Zwickau, Germany
Department of Information Science
Email: ralf.laue@fh-zwickau.de

Frank Hogrebe

Hessische Hochschule für Polizei
und Verwaltung Wiesbaden, Germany
Department of Public Administration
frank.hogrebe@hfpv-hessen.de

Boris Böttcher, Markus Nüttgens

University of Hamburg, Germany
Email: boris.boettcher@uni-hamburg.de,
markus.nuettgens@wiso.uni-hamburg.de

Abstract—The visual syntax of modelling languages can support (or impede) the intuitive understandability of a model. We observed the process of problem solving with two notation variants of i^* diagrams by means of an eye-tracking device. The number of wrongly answered questions was significantly lower when the alternative i^* notation suggested by Moody et al. was used. For the eye-tracking metrics “time to solve a task” and “number of eye fixations”, no such significant result can be given. Furthermore, we identified a deficiency for the “dependency” symbol in the alternative notation.

I. INTRODUCTION

One of the key challenges in requirements engineering is to achieve efficient communication between requirements engineers and domain experts. Several graphical languages have been developed to support this communication. In the early phases of the RE process, the language i^* [1] is frequently used for modelling the social relationships between actors.

Based on the theoretical framework “Physics of Notations” [2], Moody et al. suggested improvements on the traditional visual syntax of i^* models in order to increase its understandability [3]. To compare the understandability of the alternative notation to the traditional one, we performed an experiment using eye-tracking device. This method can be used to measure how subjects interact with diagrams (see [4]).

II. EXPERIMENTAL SETUP

For our experiment, we selected two i^* models. For both we created a version using the traditional i^* notation and a version using the visual symbols suggested by Moody et al. One model showed the classic meeting scheduler problem [1], the other model was an adaption of a model comparing possibilities for technical support to the youth counseling process [5]. Fig. 1 shows both versions of the latter model.

16 participants (8 for each notation variant) had to solve a set of six tasks while their eye movement was recorded. The participants have been recruited from students and the academic staff at the department of business information science at the University of Hamburg. Before the experiment, nobody of them was familiar with i^* . As a first step, each participant got an introduction to i^* by means of a self-learning computer course. Two different variants of the course have been developed (one using the alternative notation, the other using the traditional notation). This way, each participant

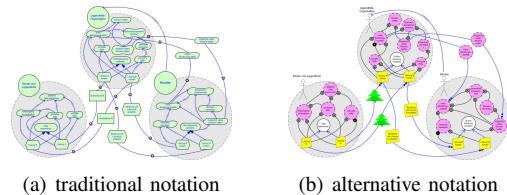


Fig. 1. Two variants of an i^* model

was assigned to the “traditional notation” group or the “new notation” group. The participants had as much time as they wished to get familiar with the i^* concepts and the notation, and they had the possibility to ask additional questions if something remained unclear.

For collecting the data, we used the eye tracking device *Tobii T60 XL* with a 24-inch screen. Using the two diagrams, the participants had to answer six questions on the screen. They were instructed to press a key as quickly as possible if they have found the answer. Afterwards, they showed their answer on the screen. This way, we collected three kinds of data: whether the answer was correct, the time to complete the task and the record of the eye movement.

Our questions were as follows: What contributes to the softgoal “Low effort”? (Q1), How many dependencies from “Autom. Appointment Scheduler” are shown? (Q2), Which resources should “Appointment Participant” provide to other actors? (Q3), If “solution A” is chosen, which softgoals of “Kids and Youth” will be positively affected? (Q4), Are there softgoals of “Youth Counseling Organization” to which selection of “solution B” has a negative effect? (Q5), For which actors a dependency from “Counselor” is shown? (Q6).

III. RESULTS

A. Correctness of the Answers

The mean number of errors in the group using the traditional notation was 2.625. For the group using the alternative notation, it was 1.125. We performed a one-tailed Mann-Whitney U-test on the data groups. The U value for the one-tailed Mann-Whitney test is 11 and the P-value is 0.014, i.e. the result is **significant** at the standard level of 95%.

All questions with exception of Q2 were answered better by the group using the alternative notation. Note that Q2 required

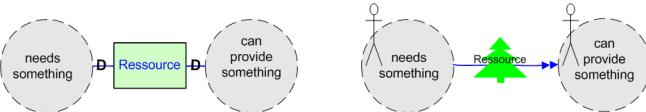


Fig. 2. Notations for Expressing a Dependency (Left: Traditional Notation, Right: Alternative Notation)

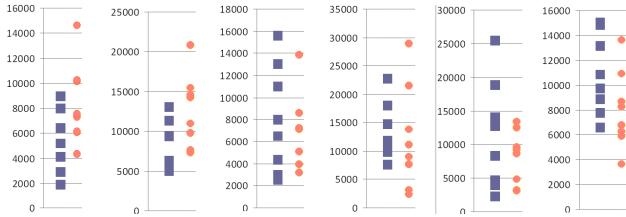


Fig. 3. Time Needed (in ms), Result for each of the Six Questions (Blue Squares - New Notation, Red Circles - Traditional Notation)

to reason about the *direction* of a dependency. As shown in Fig. 2, it is easy to misinterpret the dependency symbol \longrightarrow in the new notation in a way that a resource moves from the left actor to the right one. However, the intended meaning is that the left actor *depends on* the right actor who has to provide the resource. If we exclude the “dependency counting” question Q2, we get a **highly significant** result (P-value 0.00148).

B. Time Needed and Number of Eye Fixations

Before creating dotplot diagrams for visualising the distribution of the answering times, we removed outliers according to a commonly used method: We calculated the 25%-quartile q_{25} and the 75%-quartile q_{75} and disregarded measurements outside the interval $[q_{25} - 1.5 \cdot (q_{75} - q_{25}), q_{75} + 1.5 \cdot (q_{75} - q_{25})]$. Fig. 3 shows the distribution of the answering times to each of the 6 questions for both groups. For Q1 and Q5 (first two diagrams from the left), we observe that users using the new notation performed better. For Q3 and Q4, no differences can be observed, for Q2 and Q6 (the two rightmost diagrams) users performed a little better when using the traditional notation. While all these observations are **not statistically significant**, we observed once again that the questions for which the performance was better when using the traditional notation were the two questions that required to understand the direction of a dependency (Q2 and Q6).

From the recordings of the eye-tracker, we extracted the number of eye-fixations on the diagram while working out the answer to a question. The heatmaps in Fig. 5 visualise the durations of eye fixations when working with question Q1. A high number of fixations indicates less efficient search in the diagram. After removing the outliers in the same way as described for the variable “time needed” above, we obtained the dotplots shown in Fig. 4 where the questions are shown in the same order as in Fig. 3. It can be observed that the patterns in Fig. 4 and Fig. 3 show some similarities. Once again, **no statistically significant** differences between the groups could be found using the data shown in Fig. 4.

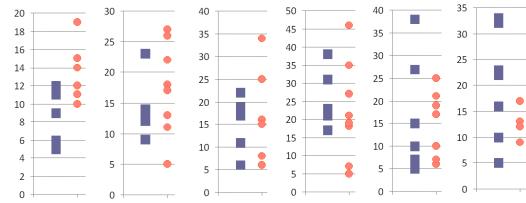


Fig. 4. Number of Eye Fixations, Result for Each of the Six Questions (Blue Squares - New Notation, Red Circles - Traditional Notation)

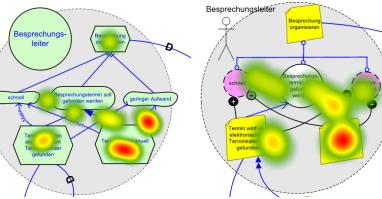


Fig. 5. Eye-tracking Heatmaps (Same Question, Different Notations)

IV. CONCLUSION

In our experiment, the number of correct answers significantly depend on the notation. This supports the hypothesis that i^* can be used more effectively when the visual notation would be improved. Answering times and number of eye fixations did not give a conclusive result. In general, we can state that the improvements suggested by Moody et al. are not always superior to the original notation. In particular, we have reason to assume that the arrow which symbolises a dependency can lead to a misinterpretation of the dependency direction. Our results are limited due to the fact that the number of participants in the experiment was rather low, therefore additional experiments will be necessary to recheck the results. The most important threats to validity refer to external validity: The subjects of our experiments (although having a background in business information science) did not have industrial experience, so the results may be different for other groups of i^* users. Second, answering questions on a screen as quickly as possible is not the typical use case for i^* models, therefore additional experiments with a more typical scenario are necessary. We are currently performing additional paper-and-pen experiments with a much larger number of participants and plan to publish a paper with full results.

REFERENCES

- [1] E. S. K. Yu, “Towards modeling and reasoning support for early-phase requirements engineering,” in *3rd IEEE International Symposium on Requirements Engineering*, 1997, pp. 226–235.
- [2] D. L. Moody, “The “physics” of notations: a scientific approach to designing visual notations in software engineering,” in *32nd ACM/IEEE International Conference on Software Engineering*, 2010, pp. 485–486.
- [3] D. L. Moody, P. Heymans, and R. Matulevicius, “Visual syntax does matter: improving the cognitive effectiveness of the i^* visual notation,” *Requir. Eng.*, vol. 15, no. 2, pp. 141–175, 2010.
- [4] A. I. Molina, M. A. Redondo, M. Ortega, and C. L. Roder, “Evaluating a graphical notation for modeling collaborative learning activities: A family of experiments,” *Sci. Comput. Program.*, vol. 88, pp. 54–81, 2014.
- [5] J. Horkoff and E. S. K. Yu, “Evaluating goal achievement in enterprise modeling - an interactive procedure and experiences,” in *The Practice of Enterprise Modeling*, ser. LNBP, vol. 39. Springer, 2009, pp. 145–160.

Symbolic Verification of Requirements in VRS System

Oleksandr Letychevskyi
Glushkov Institute of Cybernetics
Kyiv, Ukraine
lit@iss.org.ua

Thomas Weigert
Uniquesoft LLC
Palatine, USA
thomas.weigert@uniquesoft.com

Abstract—VRS (Verification Requirements Specifications) system is a tool for processing formal requirements during the initial stage of software, hardware, or system development. Symbolic modeling and deductive methods are used for detection of issues such as safety violations, deadlocks, nondeterminism, or livelocks. The formal representation of requirements also supports the generation of test suites as well as the synthesis of a design model.

Index Terms—symbolic modeling, verification of requirements, Use Case Maps, basic protocols

I. INTRODUCTION

Usage of formal methods to verify system requirements has become relevant due to the growth in complexity of the designs common in hardware and software industries. Usually verification is by a formal proof of properties of an abstract mathematical model of the system captured by the requirements. Examples of such mathematical models are finite state machines, labeled transition systems, Petri nets, or process algebras. There are two main approaches to establish properties of such models: Model checking [1] is an exhaustive exploration of the states and transitions of the mathematical model. It subsumes different techniques such as abstract interpretation [2], symbolic simulation [3], abstraction refinement [4], and others. Model checking is supported by a number of tools, such as SPIN or BLAST. The other approach is deductive verification where the requirements are presented as a set of assertions and the properties are established by theorem provers such as HOL, Z3, or Isabelle.

For the implementation of verification procedures the system requirements must be expressed formally. Some formal requirements languages are close to the programming languages (VDM, CASL). UML, SysML, and special graphical notations such as URML are also leveraged for the formal representation of requirements.

VRS [5] relies on both above approaches. Symbolic simulation in VRS allows detecting reachability of the violation of correctness properties by considering symbolic states of a system and also generates a set of tests. A static requirements checking is based on formal proving of statements about properties and involves deductive tools.

The use of symbolic and deductive methods is difficult for engineers due to the complexity and unfamiliarity of formalization and verification methods. VRS and its associated

methods aim at making symbolic techniques more amenable to industrial usage by front-ending mathematical techniques with familiar or easy to learn notations, including expressive graphical presentations.

VRS is the result of over 10 years of experience in the industrial application of formal methods to the requirement gathering stage. It was deployed in a large number of projects at Motorola and Uniquesoft LLC in various subject domains, from telecommunications to networking, microprocessor programming, and automotive systems.

II. INPUT LANGUAGE

The input language of VRS is UCM (Use Case Maps), standardized as part of URN (User Requirements Notation) in ITU-T recommendation (Z.151) which provides a scenario-based approach to requirements specification. Similar graphical approach is implemented in a tool AUTO/MAUTO (produced by Inria) [6] for specification and verification of requirements for concurrent system presented by process algebra. It can deal with model checking systems. UCM allows an easy and natural expression of sequences of events with synchronization and structure. VRS extends UCM by the language of basic protocols as developed at the Glushkov Institute of Cybernetics [7] which represents behavioral requirements as reactions of a system to externally triggered events. Such local behaviors are modeled by basic protocols which consist of three components:

- The precondition defines the state of the environment of the system at the point when the basic protocol is applicable.
- Process actions are presented as MSC (Message Sequence Chart) diagrams that show input and output signals and local actions.
- The postcondition defines the change of the environment in response to the application of the basic protocol.

Pre- and postcondition are represented as formulae of the basic logic language. It supports attributes of numeric and symbolic types, arrays, lists, and functional data types. The following example of a basic protocol (Fig.1) is taken from the specification of a well-known telecommunication protocol.

The order and synchronization of events and basic protocols is defined by means of UCM diagrams in a graphical notation. UCM diagrams are oriented graphs with initial and

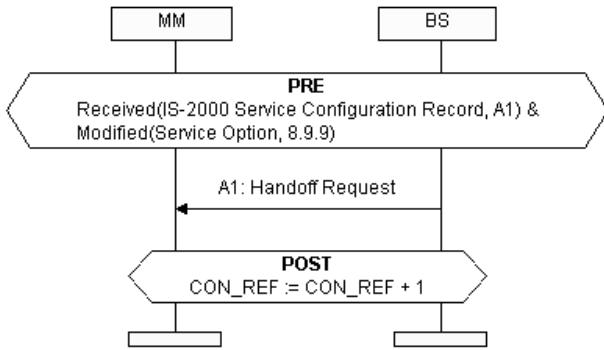


Figure 1. Basic protocol example of the requirement for a telecommunication protocol: "If an IS-2000 Service Configuration Record was received in A1: Handoff Request message and the service option was modified via section 8.9.9, the MM shall modify the attribute CON_REF by adding 1 to CON_REF"

final states. Nodes of the graph represent events in a system. The basic protocols notation captures the atomic actions (responsibilities) of a UCM map.

The UCM notation is a convenient tool for the description of parallel processes and their synchronization. An example of a UCM diagram is given below (Fig 2.).

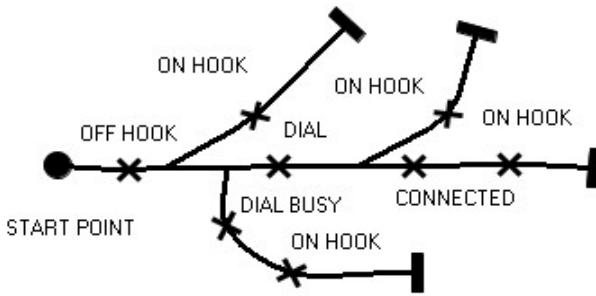


Figure 2. Example of UCM notation for requirements of phone actions in a telephony protocol

It is possible to synthesize high-level design models in terms of UML state models from these scenario models.

III. SYMBOLIC VERIFICATION

The VRS system supports the following techniques of verification of a specification.

Checking the consistency of requirements means detection of non-determinism and ambiguities in behavioral requirements. Often such issues are deeply hidden in specifications and could entail subsequent errors and misunderstanding by developers.

Incompleteness detection helps finding of deadlock situations in formal requirements. The system establishes a trace that leads to a deadlock situation and identifies its causes. A trace is graphically presented as a MSC diagram.

Safety violations are detected by means of symbolic modeling or static proving. Liveness of a system is checked by finding the reachability of the necessary property. Livelock detection identifies situations where a system may or otherwise be nonresponsive.

Different from model checking, the VRS system is based on symbolic methods that involve deductive systems such as provers or solvers. Formal requirements present the system at a high level of abstraction where deductive techniques are most suitable. Deductive systems provide proofs of assertions in a first-order theory, resultant from the integration of theories of integer and real linear inequalities, enumerated data types, uninterpreted function symbols, and queues. This technique allows the verification of the requirements for systems with a large or infinite number of states.

Symbolic techniques also support incremental verification that is important for the development of large systems. Different parts of a formal requirements specification can be verified separately and encapsulated into enlarged entities to avoid repeating the verification of such components. For a system with high degree of feature interaction each feature can be verified separately first and their interaction can be verified without examining the individual behaviors repeatedly.

IV. TEST SUITE GENERATION

The formal presentation of requirements in UCM notation together with basic protocols gives the possibility to generate a test suite at the given level of abstraction. A set of traces can be generated from formal requirements. Traces will be obtained in MSC notation and can be converted into standard test formats. Traces contain input and output signals and local actions of the system together with the set of states of the environment that contains possible values for the system attributes. These states are symbolic and cover potentially large sets of attribute values.

The generation of traces is implemented based on different coverage criteria such as requirements statement coverage or UCM branch coverage.

REFERENCES

- [1] Doron Peled, Patrizio Pellicone, Paola Spoletini, "Model Checking", Wiley Encyclopedia of Computer Science and Engineering, 2009.
- [2] Patrick Cousot, "Formal Verification by Abstract Interpretation", Lecture Notes in Computer Science, 2012, vol. 7211, pp. 3—7, Springer.
- [3] Robert B. Jones, "Symbolic Simulation Methods for Industrial Formal Verification", 2002, Springer.
- [4] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, Helmut Veith, "Counterexample-Guided Abstraction Refinement", Lecture Notes in Computer science Volume, 1855, 2000, pp. 154-169.
- [5] A. Letichevsky, A. Godlevsky, O. Letichevskyi, S. Potienko, V. Peschanenko, "The properties of predicate transformer of the VRS system," Cybernetics and System Analyses, №4, 2010, pp. 3-16.
- [6] A. Bouali, S. Gnesi, S. Larosa, "The Integration Project for the JACK Environment", Report, Centrum voor Wiskunde en Informatica, CS-R9443, 1994
- [7] A. Letichevsky, O. Letichevskyi, T. Weigert, J. Kapitonova, V. Volkov, S. Baranov, V. Kotlyarov, "Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications," Computer Networks, vol. 47, 2005, pp. 662-675.

Business Application Modeler: A Process Model Validation and Verification Tool

Sören Witt, Sven Feja, Andreas Speck and Christian Hadler

Christian-Albrechts-University of Kiel, 24118 Kiel, Germany, (swilsvfelaspelcha)@informatik.uni-kiel.de

Abstract—(Business) Process models are common artifacts in requirements engineering. The models can be enriched with plenty of (detailed) information and their at least semi formal character even enables model driven approaches or direct execution in workflow engines.

Validity of process models is crucial. Manual checking is expensive and error-prone, especially for requirements that regard the content level (e.g. compliance). To enable automated checking, an adequate method for formal specification is necessary.

We present the Business Application Modeler (BAM), which is a modeling and Validation & Verification tool that integrates modeling of processes and formal graphical validation rules. These rules can be automatically applied to process models. In particular, the modeler is supported by visualizations of checking results directly in the process models. Next to highlighting mechanisms this support includes recommendations for the correction of errors.

I. INTRODUCTION

Most kinds of manual tests are comparatively time-consuming, cost-intensive, and a high probability is given that not all errors will be detected. Moreover, such manual testing does not scale to complex BPMs with many subsystems or hierarchies. A more promising approach is to check as much requirements as possible automatically with formal methods. However, for the application of formal techniques the intended audience has to be considered. Hence, adequate mechanism for the specification of formal requirements as well as for the visualization of checking results have to be provided.

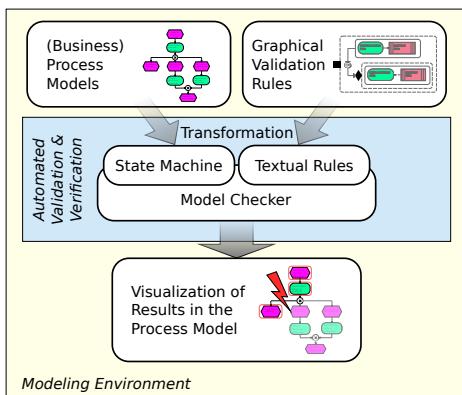


Fig. 1. Structure of BAM and the general validation process.

BAM [1] allows the graphical specification of explicit, formal requirements on the basis of (business) process models. The graphically specified requirements are used to automatically validate the models. Thereby, BAM enables the usage of

automatable and thus repeatable techniques to check various kinds of graphical process models. Figure 1 depicts BAM's core areas. First, the *modeling environment* to model processes, graphical validation rules and visualize checking results. Second, the *Validation & Verification* which enables the automated checking whether the processes models comply to the rules. In this contribution we extended BAM with mechanisms for the visualization of results beyond the highlighting of violated parts of a process like in [1] or [2].

II. BUSINESS APPLICATION MODELER

Basically, BAM is a plug-in for the IDE Eclipse. The IDE offers various basic features including a very extensible plugin mechanism. With respect to the modeling of processes, BAM comes with a basic model editor. It enables the user to define meta models for custom or existing process model notations on the basis of a generic meta meta model. For the demonstration of BAM, we utilize the *Event-driven Process Chain* (EPC) notation, which allows to enrich process models with detailed information. Basically, arbitrary properties for any model element can be specified with attributes.

A. The G-CTL notation for Graphical Validation Rules

BAM provides a rule editor that allows the graphical modeling of validation rules. The notation for these rules is the Graphical Computational Tree Logic (G-CTL) [3]. Currently, we focus on rules regarding the temporal behavior, defined in process models, although first steps with respect to structural properties have been made [4].

G-CTL allows to embed patterns of process elements (in solid rounded rectangles) into a graphical logic expression, which is a straight forward representation of the textual Computation Tree Logic (CTL). An example for a G-CTL rule is depicted in figure 2. The rule contains two patterns (in solid rectangles) and requires: If a state of the process matches the upper pattern, a state matching the lower pattern must eventually be reached. In [5] we present a detailed explanation of this and further G-CTL rules. The patterns of process elements define facts or circumstances like it is done in process models. Therefore, the rules are defined on the same level of abstraction as the process model. This makes G-CTL more comprehensible to the stakeholders of a development project than a textual CTL rule, which suffers from the indirect representation of the facts to test for.

Furthermore, G-CTL rules are specifiable in a generic manner, enabling the reusability for multiple process models.

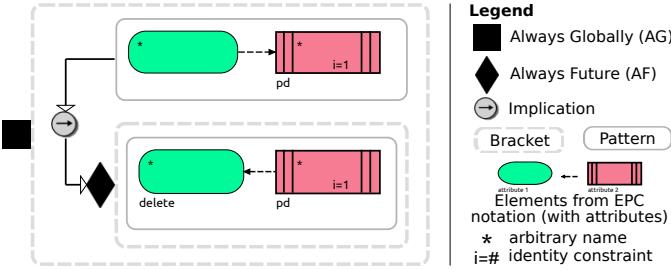


Fig. 2. A G-CTL example rule, requiring the deletion of personal data [5].

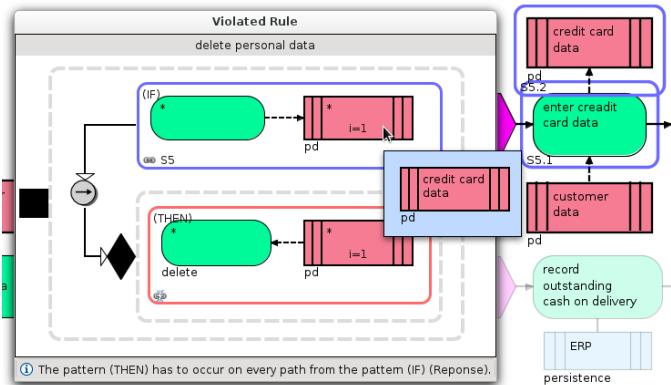


Fig. 3. Visual feedback utilizing the G-CTL rule. Elements matched by the upper pattern are highlighted. A recommendation is presented under the rule.

Instead of concrete element names wildcards ("*"), c.f. fig. 2) can be used and identity constraints (e. g. " $i = 1$ ", c.f. fig. 2) facilitate to specify if elements across different patterns refer to the same or different elements in the process. [6] provides a formal description of the pattern matching mechanism.

B. Automatic Validation and Visualization of Results

Basically, BAM allows to adapt multiple checking techniques via a plug-in interface. Currently, we prefer the formal method model checking, since it provides immediate support for CTL. The general procedure of using a model checker as checking plug-in in BAM is shown in figure 1. The process model to be checked is transformed into a textually defined state machine and the G-CTL rules are transformed into appropriate textual CTL according to [6]. After calling the model checker, BAM examines the results in order to provide visual feedback as well as recommendations to the modeler.

In case of a violated rule the model checker delivers an error trace, i. e. the sequence of states the process has gone through. Additionally, BAM evaluates if and how the patterns in the rule are matched and recognizes characteristic types of rules. This information enables the following (combinable) methods to visualize errors and support the modeler:

1) Parts of the process that did not appear in the error trace are displayed transparently, which emphasizes the defective way through the process.

2) The modeler can move stepwise through the states of the error trace. All elements that were active in the particular state are highlighted (including elements on parallel paths).

3) To help understanding how a rule affects a process, the violated rule can be displayed beside the process model (c.f. fig. 3). BAM indicates if a pattern of the rule could be found in the process model and if so, the matched elements are marked in the process, and are shown by placing the mouse over the element in the pattern (box below the mouse cursor in fig. 3).

4) BAM recognizes certain types of rules according to [7] and gives recommendations how to solve the problem—yet without any interpretation of the CTL.

C. Process Model Views – MultiView

Additionally, a concept called *MultiView* is implemented in BAM. *MultiViews* allow the assignment of specific responsibilities in the process modeling workflow and provide mechanism to handle the increasing complexity of process models. The mechanism are based on configurable filters, which determine whether and how specific model elements are depicted and whether they are editable. Examples are referenced in [1].

III. CONCLUSION

BAM allows the specification of formal, graphical validation rules (G-CTL) on the level of process models. Rules can be specified in a reusable manner, allowing to automatically check the process model for these properties (requirements). The checking results are mapped into visual feedback and recommendations. BAM has been successfully applied to the processes and rules presented in [8] and [5] and to several more complex internal test cases and intra-corporate process models.

Our further research will focus on the enhancement of G-CTL towards domain specific purposes. Moreover, we intend to build reusable rule sets for various domains. Another research direction is the tighter integration of *MultiViews* with the validation mechanism.

REFERENCES

- [1] S. Feja, S. Witt, and A. Speck, "BAM: A Requirements Validation and Verification Framework for Business Process Models," in *11th Int. Conf. on Quality Software*. IEEE Computer Society, 2011, pp. 186–191.
- [2] A. M. H. A. Awad and M. Weske, "Visualization of Compliance Violation in Business Process Models," in *Business Process Management Workshops 2009*. Springer, 2010, pp. 182–193.
- [3] S. Feja and D. Fötsch, "Model Checking with Graphical Validation Rules," in *15th IEEE International Conference on the Engineering of Computer-Based Systems*. IEEE Computer Society, 2008, pp. 117–125.
- [4] A. Speck, S. Witt, S. Feja, S. Feja, and E. Pulvermüller, "Integrating Validation Techniques for Process-based Models," in *Proceedings of the Eighth International Conference on Evaluation of Novel Approaches to Software Engineering*. SciTePress, 2013, pp. 246–253.
- [5] S. Witt, S. Feja, A. Speck, and C. Prietz, "Integrated privacy modeling and validation for business process models," in *Proceedings of the 2012 Joint EDBT/ICDT Workshops*. ACM, 2012, pp. 196–205.
- [6] S. Witt, S. Feja, and A. Speck, "Applying Pattern-based Graphical Validation Rules to Business Process Models," in *IEEE International Conference on Software Testing, Verification, and Validation Workshops*. IEEE Computer Society, 2014, pp. 274–283.
- [7] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in Property Specifications for Finite-State Verification," in *Proceedings of the 21st Int. Conf. on Software Engineering*. ACM, 1999, pp. 411–420.
- [8] T. Stuht, A. Speck, S. Feja, S. Witt, and E. Pulvermüller, "Rule Determination and Process Verification using Business Capabilities," in *Working Conf. on the Pract. of Enterprise Modelling*. Springer, 2012, pp. 46–60.

Handling Design-Level Requirements across Distributed Teams:

Developing a New Feature for 12 Danish Mobile Banking Apps

Lars Bruun, Mikkel Bovbjerg Hansen

Bankdata

Fredericia, Denmark

lars.bruun@bankdata.dk, mbo@bankdata.dk

Jørgen Bøndergaard Iversen, Jens Bæk Jørgensen, Bjarne Knudsen

Mjølner Informatics A/S

Aarhus, Denmark

jbi@mjolner.dk, bjb@mjolner.dk, bkn@mjolner.dk

Abstract—Bankdata and Mjølner have cooperated in the development of a new feature for 12 Danish mobile banking apps. Bankdata is the main system provider and Mjølner is subcontractor. Different teams from Bankdata have collected requirements, developed the necessary backend and middleware software, and designed the user interface. One team from Mjølner has implemented the app feature. The cooperation between the teams was centered around design-level requirements. Our contribution is to describe and discuss a number of lessons learned regarding requirements representations, requirements tools, and cooperation process; we have faced challenges, which were amplified by our distributed teams set-up. We also briefly describe a number of initiatives we have launched recently to alleviate the problems and improve the handling of design-level requirements in our future cooperation.

Index Terms—Pragmatic requirements engineering, process efficiency, “good-enough” requirements, agile and lean approaches.

I. INTRODUCTION

Bankdata provides complete financial IT solutions for the banking sector. Bankdata is owned by a number of Danish banks and has approximately 650 employees. Services include tools for account managers, automated banking and home banking solutions for desktop and mobile devices. Mjølner develops custom-made software solutions for Danish and international customers. Mjølner has expertise in development of a broad range of system types, among them mobile solutions. Mjølner has around 80 employees.

Bankdata uses Mjølner as a subcontractor for app development. Since early 2013, our companies have cooperated on making a new feature for mobile banking apps for 12 banks. The feature is called Swipp. It allows money transfer between two persons, using a smartphone and only based on knowledge of the recipient’s phone number. The first version was released to the Danish market in the summer of 2013. Since then, two

subsequent versions have been developed and released. In March 2014, Swipp has around 350,000 users.

In this paper, we describe our cooperation in the Swipp project, which has been organized in distributed teams. All authors have participated in the project, and together we represent the roles of user experience designers, business analysts, app developers, and project managers. From Bankdata, approximately 15 people spanning three teams have been involved. The project also included backend developers and software testers, in addition to graphical designers from another subcontractor. Mjølner had one single team consisting of four to six app developers and a project manager. Approximately 12,000 project hours have been used producing around 25 different new app screens.

Lauesen [4] classifies requirements in goal-level, domain-level, product-level, and design-level requirements. This project has primarily been concerned with product- and design-level requirements, and the main focus in this paper is on design-level requirements, i.e., the detailed specification of the user interface (unless anything else is specified, the term requirements refer to design-level requirements).

Most of our discussions in this paper involve the handling of design-level requirements internally between the distributed teams that together constitute the development organization (we refer to this as internal communication). To provide some background information, we also give an introduction to the general requirements process between the development organization and other stakeholders (external communication).

External communication is extensively described in the requirements engineering literature. However, we are not aware of many authors who have discussed the internal communication issue. Efficient internal communication is crucial on the long and complex path from initial ideas about requirements to realization

in the form of adequate solutions that are well aligned with the needs of the users.

The structure of this paper is: In Section II, we present the mobile banking apps and their context in general; this includes a description of the Swipp feature in more detail. Section III describes the general requirements process and the requirements artifacts and tools we have been using. In this way, Section II and Section III constitute background material aimed at giving the reader an overall understanding of the system we are considering and the process our work is part of. In the central Section IV, we list and discuss a number of lessons learned, including the improvement initiatives we have launched. The conclusions are drawn in Section V, which also includes a brief discussion of related work.

Even though our process, as we will discuss in more detail, had room for improvement, the Swipp feature was delivered on time, with few errors, and has been very well received and rated by the users.

II. THE APPS AND THEIR CONTEXT

The mobile banking apps are available on the iOS and Android platforms. They have been developed in parallel and have a history of four years. The apps are for 12 different banks; one of these is among the largest banks in Denmark.

There is only one codebase per mobile platform, iOS and Android, because it has been decided that all apps should have the same functionality. The majority of the apps vary in appearance, e.g., details like theme color, bank logo and other bank identity properties, which are easily parameterized. The app for the largest of the banks has been further individualized with a different main menu user interface, different graphical design, and more app preference pages with options for user interface customization.

A. Functionality

Frequently used features in the apps are viewing account balances, account entries, transferring money, and paying bills. Moreover, the apps have utility functions like currency lists, a currency calculator, and a function to block your credit card. In total, each app has approximately 100 different screens (about 25 of these are related to the Swipp feature).

A large and complex set of functionalities in the apps is the investment features. You can use this to track the stock exchange market, overview your securities and sell and buy securities. Exchange rates are visualized with interactive graphs.

The latest feature, Swipp, is the largest in the apps in terms of lines of source code and the number of app screens. The feature enables users to transfer money just by using the recipient's phone number. Account numbers are not needed, which is convenient as one person typically does not know another person's account number, and they tend to be hard to remember. When a payment has been registered, recipients are instantly notified about the transaction. Fig. 1. shows a Swipp payment screen from one of the apps.

In order to use the Swipp service, you have to sign up first. To sign up, the user completes a wizard and chooses which phone number to use, selects an account to assign to the agreement, and enters some additional details. Multiple agreements can be created and managed.

To make a Swipp payment, you enter a phone number, select a previous recipient, or select a recipient from your phone contacts, which are listed in the apps. Then you enter the amount to transfer.

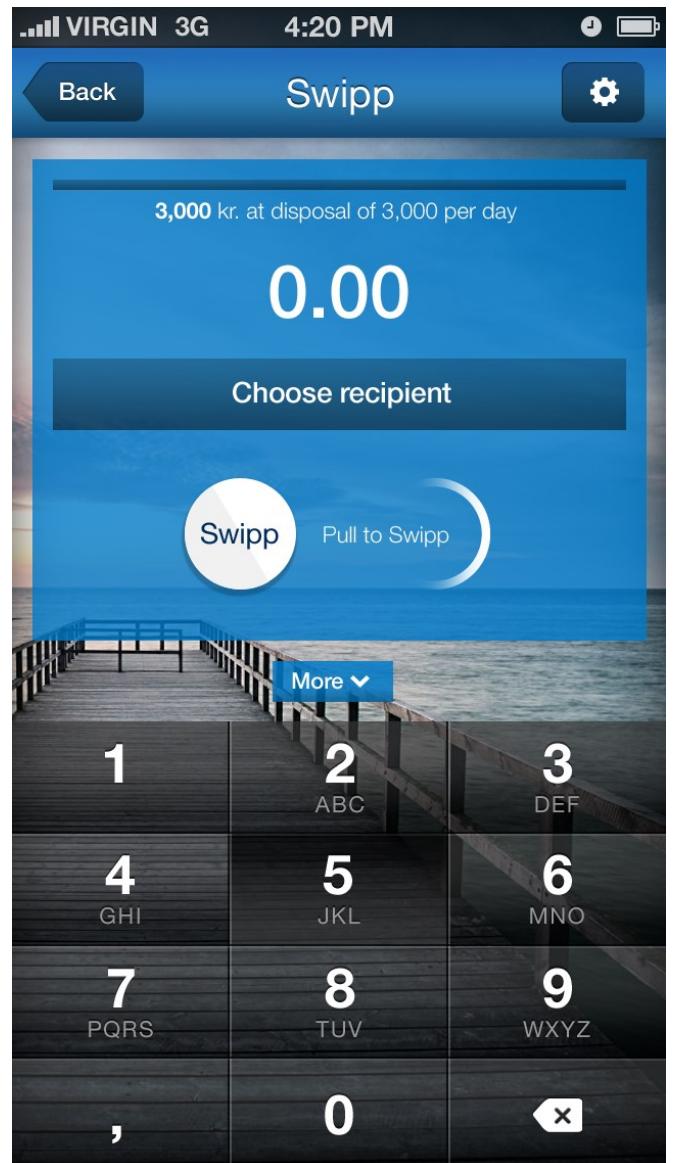


Fig. 1. Swipp payment screen from one of the apps (collapsed state).

B. Integration of the Apps with the Backend Bank Systems

All the apps integrate with the bank systems through a shared middleware interface. The middleware component is implemented in Java and has a REST interface.

The middleware interface is a borderline for division of work between Bankdata and Mjølner. Bankdata is responsible for development of the middleware including adding new service

methods and modifying existing ones to be able to handle the data requirements of the app. A team of developers at Bankdata are assigned to work with this component. Mjølner handles the development of all apps.

The middleware is a thin layer, which is acting as a facade to the backend component, which holds all the core business logic and data. The backend is also servicing other components like the home banking solutions and internal bank applications.

The backend is a COBOL component. The people working with this component are in a different team than the ones working with the middleware component. In many cases, the backend component already supported the functionality required by the mobile apps. Sometimes, however, it was necessary to change the backend component, so it could be used by the middleware to deliver an efficient service to the mobile apps.

The overall composition of the system is illustrated in Fig. 2.

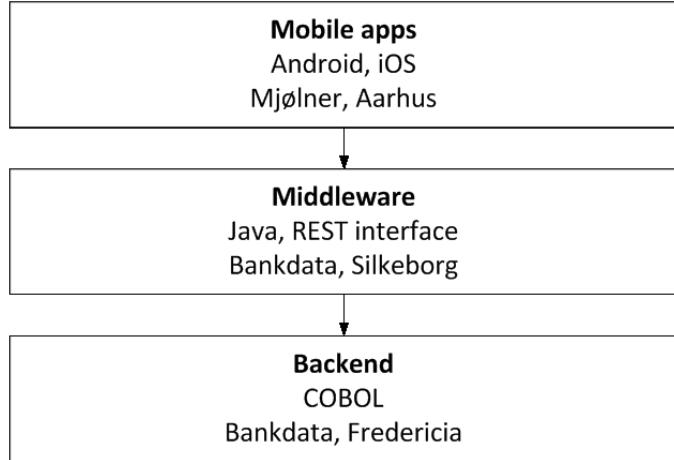


Fig. 2. Component diagram. The figure also indicates the different geographical locations with Mjølner in Aarhus and Bankdata in two cities, Silkeborg and Fredericia.

III. GENERAL REQUIREMENTS PROCESS, ARTIFACTS, AND TOOLS

A wide range of stakeholders were involved in order to specify requirements. Goal- and domain-level requirements were established by a joint venture consisting of a vast majority of the Danish banks. An example of a goal-level requirement could be that the banks should gain market shares on the very competitive market for making mobile payments; an example of a domain-level requirement could be that it should be possible to transfer money to another person just using a mobile phone.

For most parts those requirements could not subsequently be influenced by the individual banks and their IT service providers such as Bankdata. The majority of the requirements were passed on to Bankdata at the beginning of the project and remained relatively stable throughout the course of the project. All requirements from the joint venture were documented in textual form supplemented by high-level process diagrams.

At Bankdata, the project was divided into three separate project teams with project members from different departments situated at two different geographical locations. Requirements were analyzed and specified primarily in cooperation between Bankdata, the subcontractors, and various representatives from Bankdata's customer banks. Product-level requirements were typically identified at workshops with the Bankdata project team members and the banks.

After that, the requirements were incrementally detailed in cooperation between business representatives and project team members. In this way, the design-level requirements were specified. Mjølner was normally not involved at this point in the process. The requirements delivered to Mjølner were detailed and documented using:

- Use cases (prose and diagrams)
- Interaction design (wireframes)
- Graphical designs (image files)
- Middleware interface descriptions (prose)

In order to avoid multiple use case descriptions because of what was believed to be minor user experience deviations between different devices, it was attempted to keep use case descriptions generic and device independent, i.e., the same use case descriptions were used for development of both the iOS and the Android apps.

The project use case descriptions were intended to have a detail level similar to that of Cockburn's [1] fish-level description. The use cases contained, e.g., verbal and graphical descriptions of the actors involved, the main flow, and all alternative and exception flows. Use case descriptions were also supplemented with lists of functional requirements (business rules), as illustrated in Fig. 3.

| ID | Description |
|-------|--|
| FR-03 | The following input fields are mandatory: *Amount *Mobile phone number |

| ID | Description |
|-------|--|
| FR-27 | Recipient (mobile phone number) must be active in database |

Fig. 3. Extract from list of functional requirements from the 'Execute Swipp payment' use case.

An example of a flow description in the form of a UML activity diagram is shown in Fig. 4.

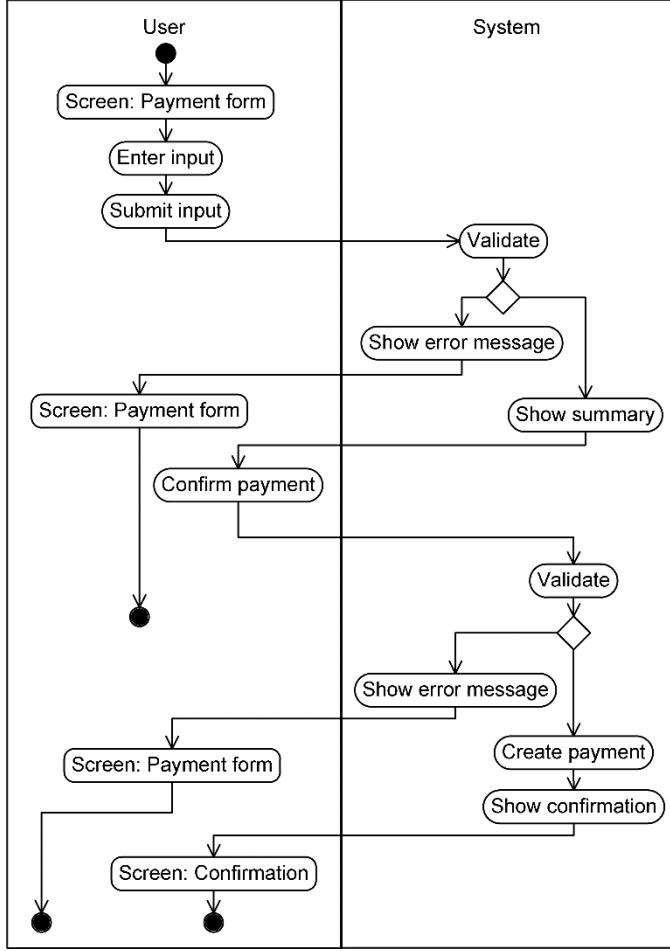


Fig. 4. Simplified version of activity diagram for the ‘Execute Swipp payment’ use case.

Interaction design [8] was documented with wireframes, which were combined into a navigation diagram including main and alternative navigation flows. An extract of the navigation diagram is shown in Fig. 5.

Wireframes and navigation diagrams were only used to document navigation flows, content, and the structural aspects of the interaction design. The look and feel was specified in the graphical design files; an example is shown in Fig. 6.

The exchange of the requirement artifacts and the parties involved are illustrated in Fig. 7. As can be seen from the figure, the following teams were involved in the requirements handling:

- Business analysis and user experience design team at Bankdata
- Middleware development team at Bankdata
- Backend development team at Bankdata
- App development team at Mjølner
- Graphical design team at another subcontractor

This distributed teams set-up was a major contributor to a number of the challenges we experienced regarding handling the

design-level requirements, as we will discuss in more detail in the following section.

Fig. 7 emphasizes the difference between external communication and internal communication, cf. the discussion in the introduction. The external communication is represented by the topmost arrow. The internal communication is represented by all other arrows.

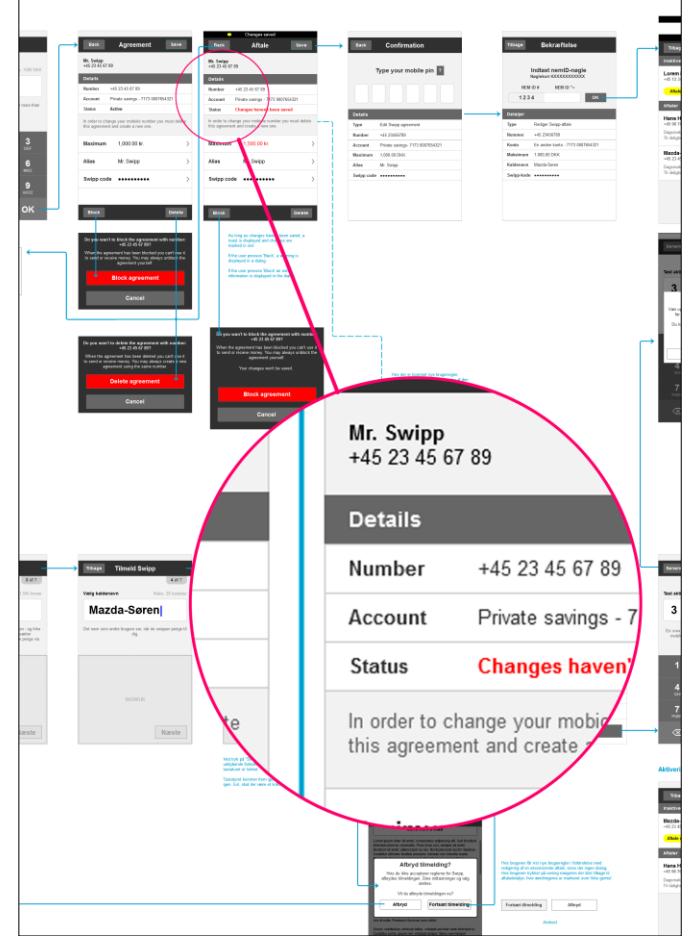


Fig. 5. Section of wireframe navigation diagram.

IV. LESSONS LEARNED

After having introduced the apps and the general requirements process, artifacts, and tools, we now list and discuss a number of lessons learned from the development of the Swipp feature.

A. Too Many Requirements Representations

The multitude of document types used for specifying design-level requirements and the distribution of those documents across several different repositories proved to be a major challenge throughout the project.

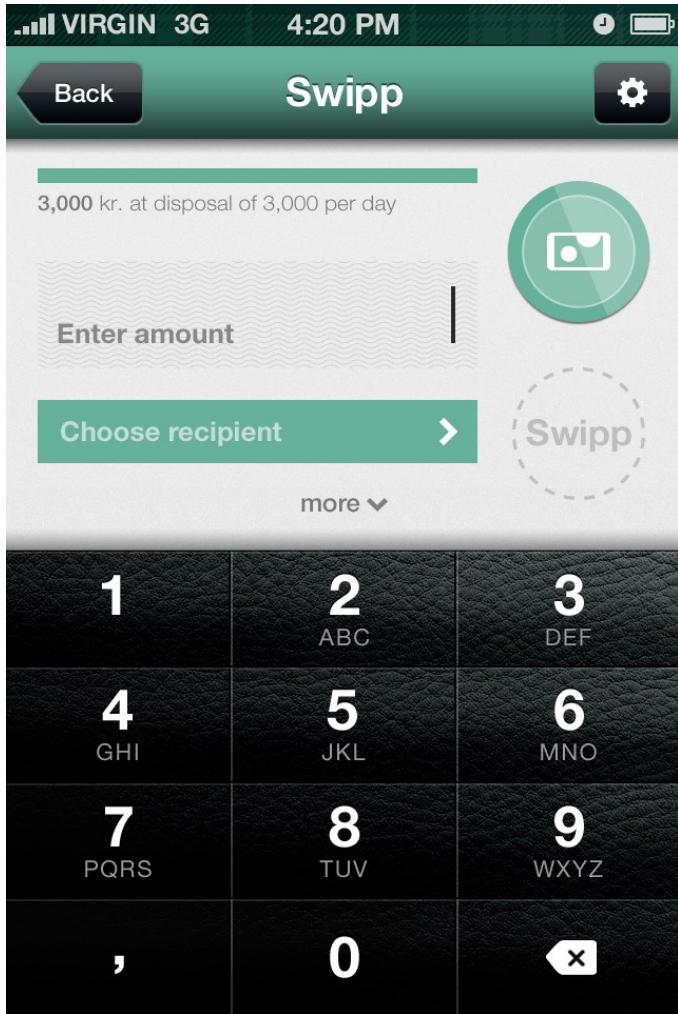


Fig. 6. Graphical design for a Swipp payment screen from one of the apps – the design varies from the design depicted in Fig. 1.

In many cases, we did not manage to avoid redundancy between the different documents. The position of an input field, e.g., would be described in four different documents: 1) a wireframe - and so would any associated label and placeholder text. In order to specify related business rules and any conditions regarding data formats and validation, the input field was referenced in 2) the use case descriptions. The visual look of the input field was illustrated in 3) the graphical design files - once again including possible labels and placeholder texts. Finally all requirements, except for those related to graphical design, would be integrated in 4) test case descriptions. As a result of this, several different documents in different repositories had to be updated because of something as trivial as the change of a placeholder text in an input field.

These conditions were further complicated by the fact that certain representations were generated from specialist tools to more viewer-friendly formats.

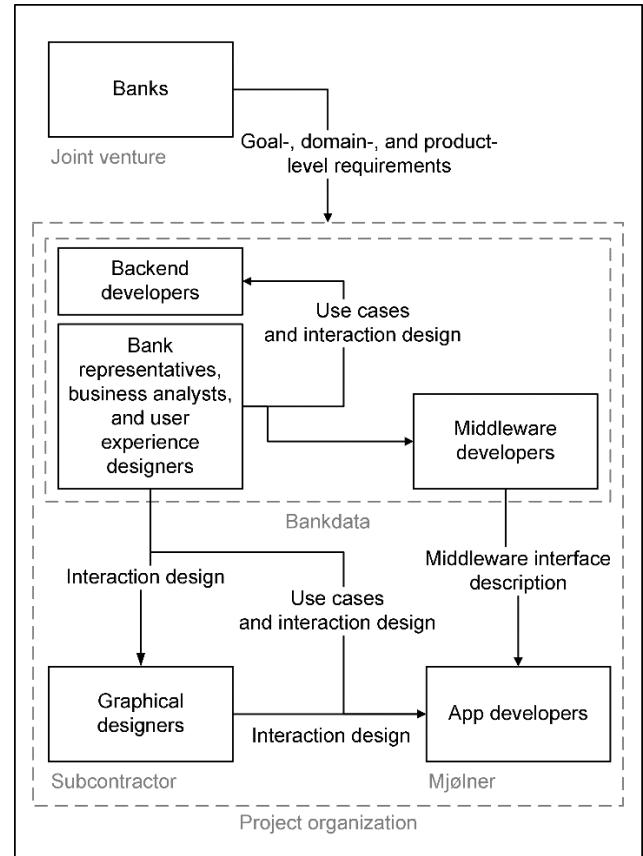


Fig. 7. Exchange of requirements artifacts and the parties involved.

Wireframes and navigation diagrams were distributed as html-prototypes, Word document specifications, or image files generated from Axure RP (a tool supporting generation of html prototypes, Word specifications, and image file wireframes). Graphical design files were distributed as image files generated from Photoshop.

This redundancy introduced a number of sources of errors and constituted a built-in inertia that hampered the process of updating and synchronizing requirements representations.

Some inconsistencies between the different representations were deliberately accepted. In order to minimize resources spent on graphical design, e.g., it was decided that graphical design files were only updated, if changes in requirements strictly involved the look and feel of a user interface element, i.e., shape, size, and tint, but not changes of phrasing, values, etc. Allowing such inconsistencies in the available documentation may seem rational and reasonable but still, however, proved to be a problem. Even when changes were made in the wireframe documents only, the set-up was not favorable. As soon as graphical design files were updated to newer versions than their wireframe counterparts, misunderstandings easily emerged. When a design file was updated (e.g., because of a new icon design) it simply proved counterintuitive to disregard the now deprecated text labels in the document, because it had a newer version date than the corresponding wireframes.

We have not been able to omit any of the different documentation types entirely. In order to eliminate the alignment

problems, we are working on reducing information redundancy. At a closer inspection, we have noticed that our use case descriptions may be too detailed. In certain aspects, the detail level is closer to clam-level than the intended fish-level, cf. Cockburn [1]. In future projects, that level of detailing will be avoided in the use case descriptions and delegated to the documentation of the interaction design in the form of wireframes.

In order to reduce the redundancy between wireframes and graphical design we have chosen that the final graphical design is primarily documented as component sheets and style sheets (depicting no real data). This way redundancy is removed and the different purposes of wireframes and graphical design are further emphasized.

B. Cooperation Process and Tool Support Were Insufficient

As described in Section III, the requirements package delivered to Mjølner contained the following artifacts: use cases, wireframes, graphical design, and a middleware interface description. The artifacts in question were created by different project members at Bankdata, which in itself poses a number of risks related to alignment of documentation.

Alignment problems, however, typically occurred after Mjølner made their initial reviews. During Mjølner's review of wireframes and use cases, mistakes and improvements were identified. Hereafter the wireframes and use cases often would get out of alignment due to insufficient communication between the different project members responsible for the relevant artifacts at Bankdata. Sometimes, this led to confusion among the Mjølner development team and the Bankdata team.

The regular communication set-up in the individual organizations could not easily be integrated with the distributed teams. Therefore, other communication tools had to be used, and since insufficient attention had been given to the communication set-up when the project was established, the needs were taken care of as they emerged. Because of that, we ended up using four different communication channels: Email, an FTP-server, a Sharepoint site, and a task management system. Mail communication in itself of course is not befitting for findability and a set-up with so many communication channels creates barriers that does not favor alignment of information.

When the implementation of the apps began and the testing of the first versions was initiated, the project needed to handle a number of requirement clarifications on a daily basis. Because the project teams were situated at different locations, the communication was primarily handled via a shared task-management system and by email and phone.

The various project teams (including the subcontractors) had several points of contact and there were no sufficiently well defined procedures for communication and documentation maintenance. When changes or further detailing of requirements were made, the original specifications were often not updated. Either because it seemed redundant or because it was unclear who was responsible for updating the documentation. Especially software testers experienced problems with keeping the test cases up to date because in many cases they were not involved

in the decision making relevant for design-level requirements and relied entirely on specifications and briefings from the other project members.

As part of Bankdata's test effort, they reported several bugs in a task management system shared with Mjølner (Jira). Some of the bugs lead to further clarification of the requirements related to the buggy feature, and this was communicated in comments to bug reports. The problems experienced with email clarification were also present when the clarification was initiated by a bug report and the discussion took place in bug report comments.

All in all, the documentation of the requirements was spread across a number of documents and repositories and distributed via several communication channels. This resulted in a number of problems: Finding the latest information about a requirement could involve traversing documents, images, emails, bug reports, and talking to colleagues. Contradicting descriptions of the same requirement could exist in the documentation, because different people were responsible for updating the documentation and because documentation was distributed across different repositories.

Much attention has been given to the future communication process that should explicitly support distributed teams. It has been decided to use a task management system as a hub for all project communication. VersionOne has been chosen since it is already implemented at Bankdata as the corporate project management system. Although documentation still resides in multiple repositories, the task management system will be the focal point for all project members. Documentation is referenced from the relevant feature groups, use cases, and tasks in the task management system, and relevant project members are automatically notified when task conversations are updated. Project members have also been asked to document decisions made at meetings or by phone in the task management conversations. Communication by email will be kept at a minimum, and it has been decided to use a shared mailbox to communicate with external stakeholders that do not have access to the task management system.

In addition to improved tool support, Bankdata and Mjølner have planned to work at the same location once a week. This is expected to enhance the communication process.

C. Software Developers Were Not Consulted Sufficiently Early

During the project, there were several incidents where implementing a component became more time consuming than expected. Often this was because the user experience designers were not sufficiently aware of the specific technical challenges in implementing particular components. We give two examples to illustrate this.

The first example involves the development of a custom component. To match the design of the app, it was decided to use a custom-made keypad instead of the operating system's built-in keypad. For outsiders it may seem like a trivial task to build a component with buttons for the digits zero to nine and a few additional characters. But for the app developers, the task was far more complex. Intricate behaviors like opening and

closing the keypad, adjusting to different screen sizes, scrolling the input field into focus, etc., often go unnoticed. All these behaviors are handled nicely by the built-in keypad, but had to be developed anew for the custom keypad. The result was that it was far more time consuming to develop this part of the app than first estimated.

Another example relates to a graphical redesign of the Swipp payment screen. At a first glance, the redesign introduced what seemed to be minor visual differences between the individual apps. But when it came to implementation, those differences added a significant overhead to the development, because many subparts of the graphical layout for the different apps had different screen positions. The developers ended up maintaining two different versions of the screen, instead of just changing style and image files. The result was more time spent on the redesign than anticipated.

In both examples, there had been a better basis for making a trade-off between app quality and effort required for implementation, if the app developers had been involved earlier, i.e., if there had been a better communication between the distributed teams. When the developers made their observations, Mjølner assumed that it was too close to the deadline to suggest changes to Bankdata, because they would have to involve the banks for approval, which would require too long calendar time. The cost of involving more people in the earlier stages of the project is likely to be less than the extra time spent in the implementation stage. Furthermore, we believe that involving the developers at an early stage could have led to more innovative solutions to certain challenges, because of the extra knowledge brought to the discussion via the developer perspective.

We have changed our cooperation process, and in future development projects the app developers will be involved as soon as business analysts and user experience designers are ready to present the first sketches of desired user experiences. The weekly workdays on the same location will hopefully ensure regular technical assessment of the desired user experience designs by app developers.

Middleware interface specifications is another subject, where app developers have been involved too late. Sometimes the middleware interface description was made after the development of the actual middleware interface. The middleware interface description was not always carefully reviewed before handed over to Mjølner, in order to verify that the middleware interface description were consistent with the wireframes and use cases.

One example of this was a set of functions involving security signing of new Swipp agreements created on the phone. In the first version of the interface, it was possible to sign an agreement, but the middleware did not support the flow through the app that the wireframes described. After some discussions between the middleware developers and app developers, a suitable compromise was reached.

One of the reasons for the lack of reviews was a result of the fact that the middleware development was not a fully integrated part of the Swipp project. Because of other assignments, the middleware developers were not allocated to the project full

time, which meant that the middleware development tasks were not always well coordinated with the rest of the project activities.

Because of this, the middleware development has now been fully integrated in Bankdata's mobile banking project organization.

D. Platform-Dependent Design-Level Requirements Were Not Considered Sufficiently Early

The iOS platform has always been the starting point when interaction design for the apps has been made. All the mock-ups, concept design and pixel perfect screens have been carefully tailored with great respect to iOS guidelines and a lot of resources have been used to make high quality designs for Apple devices.

When the iOS design was completed, the Android developers gave inputs to changes to make the Android apps less iOS-like and more in line with Android design principles. One example of this is a simple user interface control like an iOS on/off switch, which on Android should look and function more like a standard toggle button. Another example is the use of back buttons and top menu buttons in iOS, which on Android should be removed because of the hardware buttons.

The practice of designing for iOS first and using the same design for Android is very common, and is reflected in the design of a lot of the apps in the market. There are a number of reasons to do this.

Statistics for various apps, that we have been involved with, have shown much higher iOS download and traffic numbers for iOS users compared to Android users. Therefore iOS devices are often targeted first and the interaction design is then subsequently reused for Android devices in order to save resources or because the iOS design becomes an implicit point of reference for the interaction designers.

Another reason why iOS design often precedes Android design is that iOS apps have more strict submission criteria. Previously there were also considerably more design guidelines for iOS than Android, but even though this has changed, still almost anything will be allowed on the Android app market, so there is less need for tailored design on that platform.

The need for design customizations for Android apps has prior to this project been fairly manageable and adjustments were handled on the fly. In the development of the Swipp feature, however, we encountered more fundamental disparities between the interaction design suited for Android and iOS. Some of the challenges revolved around the fact that the Android ecosystem is much more fragmented than iOS in terms of display dimensions. In certain contexts, this becomes particularly important to take into account when designing.

For the Swipp feature we used new layout concepts different from the existing concepts used in the apps. An example of this is the Swipp sign up wizard, where the content of each page had been super optimized to the very limited range (currently two) of iOS display sizes. In order to draw attention to content that was hidden behind the keypad on small iOS displays, a button to hide the keyboard and make hidden content visible was introduced. This design, unfortunately, is based on a fixed layout

concept, which from an implementation perspective is not feasible when it has to be applied to a lot of different screen sizes. The latter is characteristic for the excessive amount of different Android devices.

For each of those fixed screen designs it was necessary for the Android app developers and the user experience designers to investigate and find alternative solutions to make the design work on Android. The general solution has been to handle three different groups of display sizes differently. A drawback to this solution is a more complex code base, which will require more maintenance and testing. The advantage is that Android devices with the most general screen sizes have the same layout as iOS devices.

To avoid legacy issues, e.g., with iOS designs, which do not fit well on Android devices, a couple of new approaches have been planned. In the first place, app developers for each platform will be involved in the early stages of the design process and attend meetings, where early sketch designs are presented. A more timely developer assessment of preliminary designs can ensure that layout concepts that are inappropriate from an implementation point of view are rejected. Secondly, the interaction design will be tested on multiple display sizes before it is finalized. Furthermore, the user experience designers are investigating how best practices from responsive web design can be transferred to app design in order to produce more fluid layouts that will accommodate a broader range of display sizes and aspect ratios.

These changes in the process will ensure that design-level requirements for all relevant platforms are not overlooked and also provide a better foundation for estimates.

V. CONCLUSIONS AND RELATED WORK

To sum up, the lessons we have discussed above are that 1) there were too many requirements representations, 2) that the cooperation process and tool support were insufficient, 3) that software developers were not consulted sufficiently early, and 4) that the platform-dependent design-level requirements were not considered sufficiently early.

We believe that the distribution of work across different teams has been a primary reason for the problems we have experienced. If, e.g., it had been possible to carry out the project with one single team with 6-8 persons, the problems would most likely have been less severe. In a dogmatic agile project, there are no teams distributed across different geographical locations – not even different rooms. This of course means that the needs for elaborate documentation – which may cause redundancy – and formalized communication procedures – with another balance between written and oral communication – are quite different from those of geographically dispersed development organizations such as ours. We have realized that we until recently have underestimated some of the challenges caused by the size of the project and by the distributed teams set-up.

Our project was planned to apply a number of key elements from agile development, e.g., iterations of a few weeks' length and frequent demonstrations for bank representatives. This would allow us to gain feedback that could be used to make

prioritizations for the next iterations of the project. In this way, we have in fact dealt with a number of requirements issues that have emerged throughout the project. However, there have been stages in the project, where this way of working was not fully enforced, and where feedback or other cooperative measures therefore have come too late, as we have discussed.

In addition, we have not had one explicitly appointed product owner as a central point of contact regarding requirements issues, which might have contributed to an alleviation of some of our problems. We have decided to introduce a product owner now, but we realize that such centralization may cause new problems, like introducing communication bottlenecks. Whether the advantages balance well with the drawbacks are to be seen. This remark generalizes to all the improvement initiatives, we have launched; we do not have much evidence about their effect at the time of writing this paper.

As mentioned in the introduction, we are not aware of many authors, who have described the challenges involved in handling internal communication across distributed teams in the requirements engineering literature. We do know of a couple of papers though. Gross and Doerr [3] discuss the needs for different representations of requirements for different roles. Marczak and Damian [6] set a theme which is similar to [3], and have in previous work explicitly addressed how distributed development influences cooperation patterns [2].

Regarding the lesson about not getting the software developers involved sufficiently early, it is widely recognized that requirements and software architecture are interdependent subjects that should not be treated separately or in a given, sequential order [7]. This relationship is, e.g., discussed by Loft et al [5] in the context of a project involving Mjølner, where the importance of very close cooperation between requirements engineers and software architects from the beginning of a project is described. In comparison with the Swipp project of this paper, this may be easier to facilitate when requirements engineers and software architects work within the same company and location, such that interactions do not have to cross organizational boundaries and physical separation.

A few remarks about the RE14 theme of innovation. We believe that the Swipp feature – although a rather straightforward concept – is an innovative approach to making payments. Swipp in itself is not the result of an innovative requirements process, but we think that we must focus on being increasingly innovative and creative in the ongoing requirements and development process in order to cope with the tough competition in mobile banking and mobile payments.

We hope that the improvement initiatives we have launched will be beneficial – and that they are more generally applicable and can be used in other, similar projects. In particular, we believe that projects with distributed teams must give much attention to ensuring good and efficient communication processes and strive to reduce redundancy in requirements representations.

ACKNOWLEDGMENT

We thank all our colleagues at Bankdata and Mjølner, who have contributed to the project or who have read and commented and suggested improvements to this paper.

REFERENCES

- [1] A. Cockburn, *Writing Effective Use Cases*, Addison Wesley, 2000
- [2] D. Damian, S. Marczak, I. Kwan, "Collaboration patterns and the impact of distance on awareness in requirements-centred social networks", RE07, New Delhi, India, IEEE, 2007
- [3] A. Gross, J. Doerr, "What you need is what you get: the vision of view-based requirements specification", RE12, Chicago, Illinois, IEEE, 2012
- [4] S. Lauesen, *Software Requirements - Styles and Techniques*, Addison Wesley, 2004.
- [5] M. S. Loft, S. S Nielsen, K. Nørskov, J. B. Jørgensen "Interplay between requirements, software architecture and hardware constraints in the development of a home control user interface", Twin Peaks workshop at RE12, Chicago, Illinois, IEEE, 2012
- [6] S. Marczak, D. Damian "How interaction between roles shapes the communication structure in requirements-driven collaboration", RE11, Trento, Italy, IEEE, 2011
- [7] B. Nuseibeh, "Weaving together requirements and architecture", Computer, pp. 117-117, Mar. 2001
- [8] H. Sharp, Y. Rogers, J. Preece, *Interaction Design*, John Wiley & Sons, 2007

Experience of Pragmatically Combining RE Methods for Performance Requirements in Industry

Rebekka Wohlrab
ABB Corporate Research
Västerås, Sweden
rwohlrab@mail.upb.de

Thijmen de Gooijer
ABB Corporate Research
Västerås, Sweden
thijmen.de-gooijer@se.abb.com

Anne Koziolek
Karlsruhe Institute of Technology
Karlsruhe, Germany
koziolek@kit.edu

Steffen Becker
University of Paderborn
Paderborn, Germany
steffen.becker@upb.de

Abstract—To meet end-user performance expectations, precise performance requirements are needed during development and testing, e.g., to conduct detailed performance and load tests. However, in practice, several factors complicate performance requirements elicitation: lacking skills in performance requirements engineering, outdated or unavailable functional specifications and architecture models, the specification of the system's context, lack of experience to collect good performance requirements in an industrial setting with very limited time, etc. From the small set of available non-functional requirements engineering methods, no method exists that alone leads to precise and complete performance requirements with feasible effort and which has been reported to work in an industrial setting. In this paper, we present our experiences in combining existing requirements engineering methods into a performance requirements method called PROPRE. It has been designed to require no up-to-date system documentation and to be applicable with limited time and effort. We have successfully applied PROPRE in an industrial case study from the process automation domain. Our lessons learned show that the stakeholders gathered good performance requirements which now improve performance testing.

I. INTRODUCTION

Precise and complete performance requirements are needed for software development and performance test processes [1, p. 168]. They often serve as a contract between the customer and the software supplier and have to precisely capture the customer's performance expectations (e.g., service response times). Lack of clear performance requirements can result in high costs, when unfulfilled requirements are discovered later and demand architectural changes, which are by then expensive. In general, it holds that the later a requirements-based problem is found, the higher the costs to fix it [1, p. 168].

Especially for non-functional requirements, such as performance requirements, a problem is that they are difficult to collect and describe [2, p. 194]. Despite some claims that it is “relatively easy” to specify performance requirements [1] because of their naturally quantitative nature, this is not the case in practice. Although performance metrics might be conceptually easy, it is complex to specify unambiguous and testable performance requirements. This is because they are highly context-sensitive and a lot of context information is required to specify the requirements in a precise way [3], [4, p. 5], e.g., a specification of the system's load or work situation. Furthermore, as performance requirements are always connected to concrete functions of a system, the quality of the

outcome of non-functional requirements engineering methods often depends on the quality of the functional requirement specification [5, p. 7]. The performance-specific issues come on top of general requirements engineering challenges: stakeholders often express their ideas vaguely and it is difficult to turn them into quantifiable and testable performance requirements; stakeholders are completely unavailable; functional and architectural documentations are outdated; and time and resources are scarce in industrial contexts.

In light of these challenges, we surveyed existing methods for supporting performance requirements engineering. We were looking for methods that are easy to understand, include stakeholders into the process, focus on key requirements to reduce time and effort, limit the number of missed essential requirements, result in a good basis for the creation of test scenarios, are suitable in a distributed business environment, and are applicable under time constraints. However, we were unable to identify a performance requirements engineering method fulfilling all these criteria, and decided for a combination of suitable methods.

Thus, in this paper, we present our experiences in combining existing requirements elicitation and specification methods with some extensions in a performance requirements method called PROPRE (PRactice Oriented Performance Requirements Engineering). PROPRE does not rely on the availability of all stakeholders or up-to-date documentation, is applicable in a globalized environment, and goes along with a moderate and feasible time effort. To achieve this goal, it solely relies on abstract feature models [6] of the system under study. Furthermore, we provide performance requirements specification templates to ensure that all relevant attributes of the gathered performance requirements are recorded.

We successfully applied PROPRE by collecting and specifying the performance requirements of one of ABB's large industrial size systems. We report the lessons we learned from this case study. Based on these lessons, we sketch possible variations or future improvements of requirements elicitation and specification methods.

The paper is structured as follows. In Section II, we provide background information concerning performance requirements and the context of our study. Section III presents our requirements for a performance requirements engineering method to be applicable in our industrial context. Section IV summarizes

the findings of a survey of performance requirements elicitation methods. Based on the survey, Section V presents all steps of our combined performance requirements engineering method PROPRE. We report on using PROPRE on a large scale ABB system in Section VI. We critically reflect our method in the light of this case study in Section VII and present recommendations based on our lessons learned in Section VIII. After presenting related work in Section IX, Section X concludes the paper.

II. BACKGROUND

This section introduces performance requirements and metrics (Section II-A) and the context of our study, which is the ABB device diagnostic service system (DDSS) (Section II-B).

A. Performance Requirements

Performance requirements belong to the group of *non-functional requirements* as opposed to *functional requirements*. Functional requirements describe the functionality a system provides. Non-functional requirements are commonly described as “information on or restrictions with regard to quality characteristics of the system” [7, p. 16].

Performance requirements set how the system should operate under time and resource constraints. They are connected to functional elements and specify constraints on them. For example, “99% of all response times for Task A shall be less than 2 seconds during the busy hour” [4, p. 5].

Performance requirements are specified using performance metrics. The most relevant performance metrics are [8]:

Response Time — the time the system needs to respond to a request (starting with the submittal of the request and ending with the full arrival of the system’s response)

Throughput — the amount of requests a part of the system can handle in a certain time interval or the amount of data that can be delivered, for example, over a network connection, in a certain time interval

Utilization — the usage of a resource (like the CPU) due to a part of the system, relative to its maximum capacity

Dynamic Capacity — the amount of entities a part of the system can handle simultaneously, i.e. when the utilization equals 100%

Static Capacity — the quantity of a particular type of entity that the system must be able to store permanently (typically in a database)

B. Context of the Study

In the study presented in this paper, we specified performance requirements for the ABB device diagnostic service system (DDSS). This system records status information, failures, and other data of thousands of industrial devices and provides them with different services. The system consists of more than 0.5 million lines of code and is engineered at four locations in three countries that each host a hand full of developers. The DDSS has been deployed for several years and has many customers worldwide.

The DDSS performance requirements were insufficiently defined earlier. Therefore, they should now be systematically elicited and documented. The performance requirements are needed for future development, performance testing, and monitoring of service level objectives in production. Since performance requirements are complex and context-sensitive, it is expensive to collect and precisely specify them. The DDSS owner asked us to focus our effort on aspects that had been problematic in the past. Therefore, we restricted the requirements engineering process to the four metrics: *Response Time*, because ABB’s customers should be served swiftly; *Throughput*, to set efficiency goals that help us handle all connected devices; *Utilization*, because DDSS services share key resources and should not starve each other; and *Dynamic Capacity*, to define what device failure peaks the DDSS must handle. Furthermore, we limited the number of requirements to specify based on our time budget.

The following four characteristics of the system make it difficult to apply existing requirements engineering methods. First, little complete up-to-date documentation exists that can be built on when eliciting requirements. Second, the stakeholders have little time for the requirements engineering process. Several stakeholders are responsible for multiple products and are thus only part-time available to DDSS tasks. Third, due to the distribution the DDSS teams it is difficult to schedule meetings in which all stakeholders can participate. Fourth, the effort for the requirements engineering process has to be minimized for cost efficiency.

III. CRITERIA FOR PERFORMANCE REQUIREMENTS ENGINEERING METHODS

We created a set of criteria for a performance requirements engineering method to be applicable in our industrial environment. We developed the criteria based on experience that we gained in earlier projects at ABB. We expect that these criteria are equally applicable in other companies.

Our performance requirements engineering method had to

- C1) Be easy to understand for stakeholders without long explanations, i.e. for both technical and management stakeholders/non computer science experts.
- C2) Include stakeholders into the process and encourage discussions.
- C3) Help to focus on key requirements to reduce time and effort.
- C4) Result in a good basis for the creation of test scenarios.
- C5) Be suitable in a distributed business environment
- C6) Be applicable under time constraints

IV. SURVEY OF ELICITATION METHODS FOR PERFORMANCE REQUIREMENTS

Because we consider *Requirements Elicitation* the most essential and difficult phase of the requirements engineering process, we analyzed different elicitation methods for their applicability in our context. Requirements elicitation is highly relevant because the outcome of this phase determines the

TABLE I
SUMMARY OF ANALYSIS OF THE ELICITATION METHODS

| | Surveyed Methods | | | | |
|--|-------------------------|-----------------------------|---------------------|----------------------------------|-------------------------|
| | NFR framework [9], [10] | Annotated feature model [6] | The NFR Process [7] | Quality Attribute Workshops [11] | Misuse Case method [12] |
| C1 Understandable for stakeholders | - | + | + | + | - |
| C2 Includes stakeholders, encourages discussion | - | + | o | + | o |
| C3 Focus on key requirements | o | + | - | + | o |
| C4' Easily specifiable and testable requirements | - | - | + | + | - |
| C5 Suitable for distributed environment | o | o | - | - | o |
| C6 Applicable under time constraints | o | + | - | o | o |

completeness of the resulting set of requirements. Since all follow-up phases are based on requirements elicitation, errors made in this phase cannot be easily corrected later.

We conducted a survey of elicitation methods for performance requirements using the criteria from Section III. An additional criterion is that the method should not only be a good basis for test scenarios, but shall also ease the requirements specification. Thus, we extend criterion C4 to “C4’ Result in a good basis for the requirements’ specification and the creation of test scenarios”. The resulting set of criteria forms the rows of table I.

We analyzed five methods using the criteria stated above. The methods are located in the columns of Table I. The methods were selected based on a systematic literature review. They represent different types of elicitation methods for non-functional requirements. Our detailed survey criteria are discussed in [13].

The *NFR framework* [9], [10] uses goal graphs to derive implementable requirements from high-level quality goals. In layers, the quality goal, e.g., performance, is decomposed step-wise to lower level goals that satisfy the higher level goals. While the step-wise refinement appears intuitive, Chung and Nixon report that it may not be easy to understand for all stakeholders [14].

The *annotated feature model* method [6] is based on a hierarchical model of the system’s features. A feature is a unit of functionality which is implemented by at least one sub-system of the system. To collect non-functional requirements, quality metrics are attached to features. This structured and time-efficient approach helps to ensure most system parts and functionalities are covered by non-functional requirements. The annotation technique works well when only a limited set of metrics are considered.

The *NFR Process* method [7] is based on an algorithm that iterates over all functional elements gathering relevant quality attributes for them. In elicitation interviews, a pair-wise comparison is conducted selecting appropriate quality metrics for each of the functional elements. A weakness is that the complexity of the pair-wise comparison explodes for realistically sized systems.

Quality Attribute Workshops [11] are structured group meetings to elicit and specify functional and non-functional requirements by collecting, refining, and prioritizing scenarios. Such workshops are accepted industry practice to collect general non-functional requirements. However, they are not suitable for a distributed environment in which stakeholders cannot be brought together.

The *Misuse Case* method [12] is typically used in the context of security requirements. It collects information about scenarios that should be avoided and can be used to misuse the system. The terminology that is used by the method (e.g., vulnerability, threat, countermeasure) is unintuitive in the context of performance requirements. When it comes to performance requirements, we deal with the system in its normal state whereas Misuse Cases are focused on exceptional conditions of the system. That makes the method difficult to apply without adaptations.

As a result of this analysis, we decided to use the feature model method for requirements elicitation because it fulfills most criteria, but to combine it with suitable methods for the remaining requirements engineering phases.

V. THE PROPRE METHOD

This section presents the PROPRE (PRactice-Oriented Performance Requirements Engineering) method, which combines the annotated feature model for requirements elicitation [6], an importance and difficulty ranking, and an adjusted Volere template [15] for requirements specification with standard activities for discovering needs and for validation [16]. Figure 1 presents an overview of the steps in our method and the artifacts produced and consumed by each step. The background panes indicate the requirements engineering phases that the steps cover.

In the following, we outline this process. In step 1, we collect background information connected to the system which results in lists of stakeholders and features (Section V-A). Based on the list of features and input from the stakeholders, we elicit the requirements by creating a feature model in step 2 (Section V-B). The elements in the model are annotated with performance metrics and ranked based upon the importance of performance and the difficulty of achieving performance goals. The requirements are specified and refined in several iterations in step 3 (Section V-C) using adapted performance specification templates. At the end of the requirements engineering process, the specified requirements undergo final validation in step 4 (Section V-D).

We give details on each of the four steps in Section V-A to Section V-D.

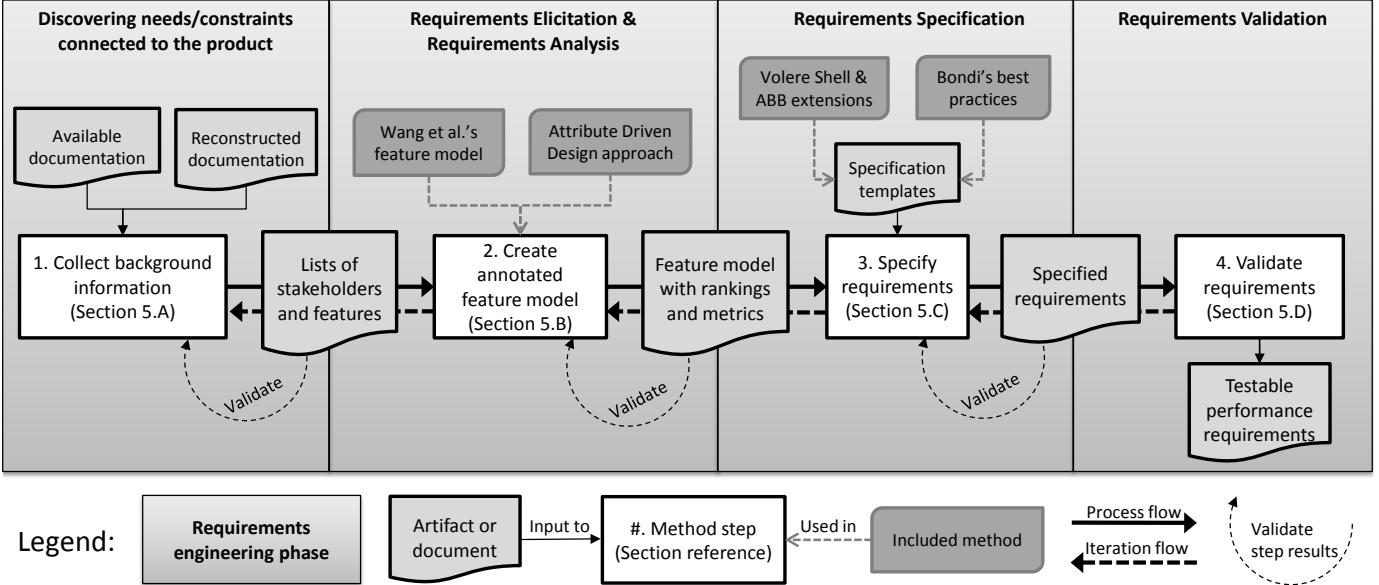


Fig. 1. The steps of the PROPRE method

A. Collect Background Information

In this step, we gather background information about the system and its stakeholders. One way to do this is to model all involved stakeholders in a network diagram to visualize the connections between them and to get an overview about whose interests need to be considered. We collect information on the functionality of the system and create a list of features using available documentation.

Documentation is often outdated or incomplete in practice. Our method does not require full, or likely any, reconstruction of information in this step. The goal is to construct an initial list of features to expand and review during the next step. As we plan for several iterations of the feature model creation step, the list of features can be incomplete or outdated at this point.

B. Create an Annotated Feature Model

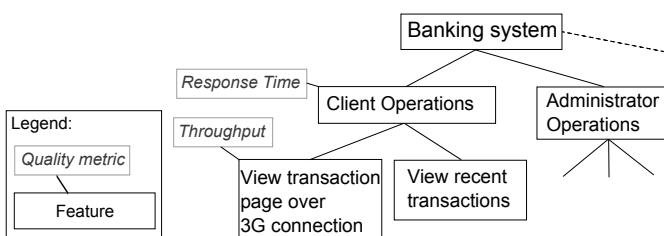


Fig. 2. Example of an annotated feature model

For the requirements elicitation, we extended Wang et al.'s [6] annotated feature model with an importance and difficulty ranking for each feature. To start the model creation, a meeting with available stakeholders is scheduled. In case key stakeholders are unavailable, they can be contacted in a later step.

| Client Operations | |
|-------------------|-------------|
| Importance | Difficulty |
| H (M) L | (H) M L |
| Throughput | Utilization |
| Response Time | Capacity |

Feature Name
Rankings
Performance metrics

Fig. 3. Example of a filled-in feature sheet

Basic Annotated Feature Model A feature model is a hierarchical model in which a system is broken down into subsystems, components, and features. Figure 2 shows an example of an annotated feature model. The features (boxes with black border) are annotated with quality metrics (boxes with grey border and italic text) to collect non-functional requirements.

The first step in the meeting is to complete the feature model itself. This can be seen as an iteration of the first step “Collect background information”. In case the available documentation in step 1 was not complete, it is important to identify and reconstruct missing features together with the participants. In case we find that features are outdated or unnecessary, we rename them and remove irrelevant features. We need to collect all relevant features to ensure that we can identify all relevant performance requirements later on.

We suggest to prepare paper sheets for the features. Each sheet has a section for the feature name and space for the rankings and performance metrics. Figure 3 shows an example of a filled-in feature sheet. These *feature sheets* can be pinned to a whiteboard, for example, to construct a feature model.

Importance and Difficulty Ranking During the second step in the meeting, we conduct importance and difficulty

rankings of the features to be able to select which requirements to specify later. Specifying requirements only for the most relevant features limits the effort needed to apply the method. This ranking task belongs to the *Requirements Analysis* phase.

The *importance* ranking should reflect the impact of a feature's performance on customer satisfaction and the saleability of the product. For the *difficulty* ranking, we considered the development and test efforts needed to improve and to measure the feature's performance.

Our ranking step was inspired by the *Attribute Driven Design* [17] approach. In the context of ADD, the stakeholders are asked to conduct a ranking of specified requirements according to their importance and their impact on architectural decisions. We decided to conduct a similar ranking for the features, according to their importance and difficulty. As suggested for ADD [17, p. 15], we use the simple ranking values *High (H)*, *Medium (M)*, and *Low (L)*. Setting quota on how many features should be ranked High, Medium, or Low may help to balance the ranking.

Performance Metrics The third meeting step is to select performance metrics for the features. For each feature we ask the participants to suggest and discuss appropriate metrics. The selected metrics can be underlined on the feature sheets. Structuring the discussion with concrete features and performance metrics helps to discover stakeholder needs. Iterating over all features and selecting relevant performance metrics ensures that all relevant performance requirements for the system's features are identified. Of course, the importance ranking may be used to prioritize what features to discuss and thereby limit the time for the discussion.

Refinement and Validation After the feature model has been created, we refine the feature model and validate our preliminary results. We contact stakeholders which were not part of the feature model creation meeting to review and refine (selected parts of) the model. The feature model is ideal for this iteration cycle, because no time has been spent yet on the details of the requirements. As new information comes in it is easily integrated into the model to update the overview picture. The refinement stage is especially useful if key decision makers cannot participate in the model creation meeting. Once the revisions stabilize, we contact all stakeholders to confirm the correctness of the model and move to the next step.

C. Specify Requirements

The *specification* phase of the PROPRE method is based on adapted performance requirement specification templates described in Section V-C1. Afterwards, the specification process is described in Section V-C2.

1) *Specification Templates*: We propose performance requirement specification templates that are an extension of the Volere Shell [15]. The Volere Shell is a general requirement specification template that captures the core properties of a precise requirement. ABB had previously extended the Volere Shell with performance specific details for internal use. We have taken these already extended templates and added in-

TABLE II
TEMPLATE FOR RESPONSE TIME REQUIREMENTS. OUR EXTENSIONS TO THE VOLERE SHELL ARE SHOWN IN *italics*.

| Requirement# | Requirement Type | Event/Use Case# |
|--|------------------|---|
| Description | | |
| <ul style="list-style-type: none"> ○ Operation Type ○ <i>What operation is this requirement connected to?</i> ○ <i>Where does the operation start and end?</i> ○ <i>Involved components</i> | | |
| Fit Criteria (constraints on operations/components) | | |
| <ul style="list-style-type: none"> ○ <i>Value for metric</i>, e.g., <i>tolerable response time</i> ○ <i>Timing boundary start and end</i> ○ <i>Degree (in %) to which the requirement has to be fulfilled</i> ○ <i>The size of the time interval in which the requirement is measured</i> ○ <i>Hardware setup and constraints</i> ○ <i>Condition of the system</i> ○ <i>What assumptions about the rate at which the operations occur are made?</i> | | |
| Rationale | | |
| <ul style="list-style-type: none"> ○ <i>Justification of values or quantities</i> ○ <i>Exceptional case: rationalize when requirement does not have to be fulfilled</i> ○ Motivation for the requirement's existence | | |
| Source | | |
| Who/what raised the requirement | | |
| Customer Satisfaction when implemented | | Customer Dissatisfaction if not fulfilled |
| Dependencies | | |
| <ul style="list-style-type: none"> ○ List of requirements that impact/influence this requirement ○ <i>List of dependent requirements</i> | | |
| Supporting Materials | | |
| History | | |
| <ul style="list-style-type: none"> ○ <i>Name of a subject matter expert on this requirement</i> | | |

sights from Bondi's "Best Practices for Writing and Managing Performance Requirements" [4]. Bondi presents patterns and anti-patterns for unambiguous, precise, and traceable performance requirements.

The response time requirement template is shown as an example of our extensions in Table II. It captures all relevant core properties of a requirement, grouped in several categories (e.g., "Fit criteria"). In the table, our extensions to the Volere Shell are printed in *italics*. Our changes do not touch the structure of the Volere Shell, but force the specification of more precise requirements. An example of a filled-in template is shown in Table III. It can be seen that the structure is taken from our template (Table II) and filled with concrete values.

We expanded the description field with queries for context information, such as involved system components, that frames the requirement description. We foresee that a performance requirement may cover several use cases or only part of a use case. Therefore, we ask what the start and end of the operation is for which the performance requirement will be defined.

The fit criteria field saw most expansion. First, of course, a value should be stated together with a clear definition of the measurement points (start and end). Second, it has to be described to what degree the requirement needs to be fulfilled. This can be specified as the percentage of the measured cases that have to meet the requirement and a measurement interval. For example, the degree could be the average case or 99% of the measured cases. The time interval could for example be an

TABLE III

EXAMPLE OF A FILLED-IN REQUIREMENT SPECIFICATION TEMPLATE FOR RESPONSE TIME

| Requirement# | Requirement Type | Event/Use Case# | | |
|--|---------------------------------|-----------------|--|--|
| 35 | Performance | 10 | | |
| Description | | | | |
| Operation Type. Any inquiry shall complete the display of its results | | | | |
| Fit Criteria | | | | |
| <ul style="list-style-type: none"> ◦ (Tolerable length of time) In no longer than 4 seconds ◦ (Timing boundary start) From the time the user submits the request ◦ (Timing boundary finish) Until the system displays the results ◦ (Degree of fulfillment) For at least 99% of the inquiries submitted each hour ◦ (Indicative hardware set-up) When using a 1Mbit DSL internet connection ◦ (Condition of the system) While the system serves 200 simultaneous users, 5000 unique users per hour and system replication is copying the system state. | | | | |
| Rationale | | | | |
| <ul style="list-style-type: none"> ◦ Justification of values. This figure is based on acceptance tests of previous version of system indicating that users begin to lose patience soon after this time. ◦ Exception Case (High load caveat). This requirement does not apply to inquiries across large volumes of data where arbitrary selection criteria are allowed ◦ Motivation. To ensure customers do not lose patience waiting for the system's response | | | | |
| Source | | | | |
| Historical acceptance tests of previous version of system | | | | |
| Customer Satisfaction | Customer Dissatisfaction | | | |
| 4 | 4 | | | |
| Dependencies | | | | |
| None | | | | |
| Supporting Materials | | | | |
| Acceptance test results of previous version of system | | | | |
| History | | | | |
| This requirement was first raised by Product Manager on 12/02/2008 | | | | |

hour. An imprecise statement is “The requirement has to be valid 99% of the time”. It is better to say that the requirement must be fulfilled 99% of an hour or 99% of a year [4, p. 5]. Third, the system context assumptions have to be specified. Performance is clearly dependent on the hardware setup the software is executed on, what state the system is in, and what other workloads exist on the system.

The rationale field was updated to include a justification for the value set for the performance metric. For example, the justification in the example in Table III is that acceptance tests have shown that customers start to lose patience after a certain amount of waiting time. The justification should answer the question why a particular value was set, whereas the motivation for the requirement explains why a performance requirement was set. As part of the fit criteria, a fulfillment degree was specified. In the rationale field, we leave room to specify what exceptional cases cause the fulfillment to be less than 100%. Here one can also note what could violate the assumptions on the condition of the system.

We suggest to add a trace of dependencies to/from other requirements (bidirectional traceability) in the dependencies field. This is especially helpful for later revisions of the requirement, so that connected requirements can be easily identified and changed as well. The name of a subject matter expert is suggested to be added to the history field.

2) *Specification Process:* The specification can be done by the requirements engineer, but ideally it is done in a meeting with selected stakeholders. If a meeting is scheduled, the requirements engineer can and probably should prepare the templates based on the knowledge gathered while constructing the feature model.

If the requirements are specified together with the stakeholders, the first step is likely an introduction on how to

write performance requirements on the templates. It should be explained to the stakeholders that the resulting requirements need to be unambiguous and precise to form a good basis to create test scenarios. It is important to evaluate the requirements together with the stakeholders and ensure that these characteristics are fulfilled.

The second step in a meeting depends on the size of the group. If the group is larger, it can be broken down into smaller groups. Each of the sub-groups then does an initial specification before reviewing all results in the larger group. Alternatively, the requirements can be discussed and specified together with all participants.

After the specification meeting, we refine the requirements together with key management stakeholders and customers. We show them our preliminary results and complete the requirements where information is missing. Face-to-face meetings, but also telephone conferences can be scheduled, so that stakeholders in different locations can be contacted. We go through the requirements, review, and refine them.

D. Validate Requirements

Throughout the requirements engineering process it is useful to work in iterations. We contact the stakeholders at regular intervals (for example, once a week) to validate that we elicit and specify all relevant and only important requirements.

During the final validation, we distribute the specified and refined requirements to all involved stakeholders. We ask our stakeholders to validate the requirements’ correctness. We ask the test team to review the requirements for testability. In case the result is not satisfying, we deploy iterations to improve it.

VI. CASE STUDY

In this section, we present the results of applying the PROPRE method on ABB’s device diagnostic service system (DDSS) (cf. Section II-B). We first discuss each step of the method in turn: we describe how we collected background information (Section VI-A); what the results of the feature model creation were (Section VI-B); how we specified the requirements (Section VI-C); and how we validated the requirements (Section VI-D). Afterwards, Section VII discusses strengths and weaknesses of PROPRE that we learned by applying the method in our case study.

A. Collecting Background Information

We started the requirements engineering process by collecting involved stakeholders and their connections. This step was easy because we collaborated with several stakeholders before. We identified the product management (Sweden), development teams (Sweden 2x, France), and test team (India) as the key stakeholders. All stakeholders know each other, but not all have met face to face.

We used available documentation to get an overview of the DDSS and to construct an initial list of the system’s features. There was only little and outdated documentation available which made it necessary to complete and adjust our list of features later. We had sufficient trust in that the feature model

creation would help us complete the feature overview. We could thus avoid spending time on reconstructing information.

B. Requirements Elicitation Using a Feature Model

We deployed the feature model exercise in a 90 minutes face-to-face meeting in which stakeholders from all groups participated. This small amount of time was enough to complete the feature model and attach rankings and metrics to the features. The following paragraphs summarize our findings during the elicitation step with the feature model.

Dealing with Incomplete Information We had based our initial feature list on limited and outdated documentation. We asked the participants to complete the list of features and gathered all relevant features, but ended up with some redundant features and some features with unclear or incorrect names. As the feature model does not define functionality, it was sometimes unclear what a feature name covered. For example, the feature name *process error upload* had to be refined to make the feature's scope clearer.

Outcome of the Ranking Exercise The importance ranking exercise was difficult to conduct during the meeting due to absence of the product owner. He has the best knowledge of the feature priorities. Thus, for some rankings, we set estimated guesses as suggested by the participants.

The difficulty ranking was easier to conduct. The development team could make reasonable estimates of the implementation effort needed to fulfill a particular performance requirement.

Unavailability of Key Stakeholders The product owner, an important stakeholder, could not participate in the meeting. However, with the feature model, we could proceed iteratively and add new information later. We scheduled a separate meeting with the product owner and successfully integrated his viewpoint.

The product owner had a very clear idea of the DDSS's features and business drivers. Therefore, we needed just 45 minutes to review the feature model and update priorities. We had printed the model on a large sheet which turned out to be very helpful as a basis for discussions. We could easily annotate the feature model directly on paper.

Use Case Diagrams to Map Actors We did not have documentation on the system's actors available, but which actors use a feature can impact requirement priorities. The business impact of different actors and their satisfaction differs. For example, end customer web requests might require quicker answers than those on the internal website. Therefore, we decided to iterate over the background collection step and reconstruct a basic use case diagram to document actors.

The feature model helped us to target the reconstruction effort. The features were a fair starting point for use cases and we could be selective about what features to cover. After constructing the use case diagram we could make a better trade-off for feature importance rankings with the input obtained from the product owner.

The Created Feature Model Parts of an anonymized version of the feature model created in the feature model creation and refinement meetings are shown in Figure 4. The rankings' results are shown in pairs (*importance ranking*, *difficulty ranking*) below the features' names. The figure shows four subsystems. Below each of the subsystems the related features are modeled. Some features belong to more than one subsystem which we indicated by positioning them in the middle of two subsystems. For example, Feature F5 belongs to both Subsystem S1 and S2.

Refinement of the Feature Model We created a digital version of the feature model that we sent to the stakeholders at different locations for review. The stakeholders could annotate the feature model and give us their feedback. The digital version of the feature model improved our method's applicability in a globalized environment.

Based on the follow-up meetings and e-mail feedback, we adjusted the feature model and introduced new features, renamed and changed others, and split some. If the name of a feature is called F_i' in Figure 4, it has been changed in the refinement step and was initially F_i .

C. Requirements Specification

While creating the annotated feature model, we had already found that we would need more information about the DDSS and its functionality. This led us to reconstructing use case diagrams to identify the system's actors. The first step we had to take during specification was to take another step back to iterate and create short feature descriptions. Without these descriptions it is difficult to specify requirements in sufficient detail. We did this by extending the previously created use case diagrams with scenarios. Creating the scenarios helped, for example, to decide on the start and end points of response time definitions.

Then, we selected features and their associated performance metrics from the feature model based on their ranking. For each selected feature-metric pair, we filled in a performance requirement template. We limited the specification step to 20 requirements to bound the effort needed. With this number, we could include all high priority features.

To specify the requirements, we met with the development team that is responsible for the technical requirements specification. Together, we filled in the specification templates. We again changed some features' names and clarified scenario descriptions during the specification. We also occasionally switched metrics or dropped metrics that we found did not make sense.

After the requirements specification meeting, we shared the results with all stakeholders. Then, we refined the requirements together with different stakeholders. For instance, we scheduled a telephone conference with one of ABB's internal customers. Based on the customer's input, we refined scenarios to focus on the customer's performance experience.

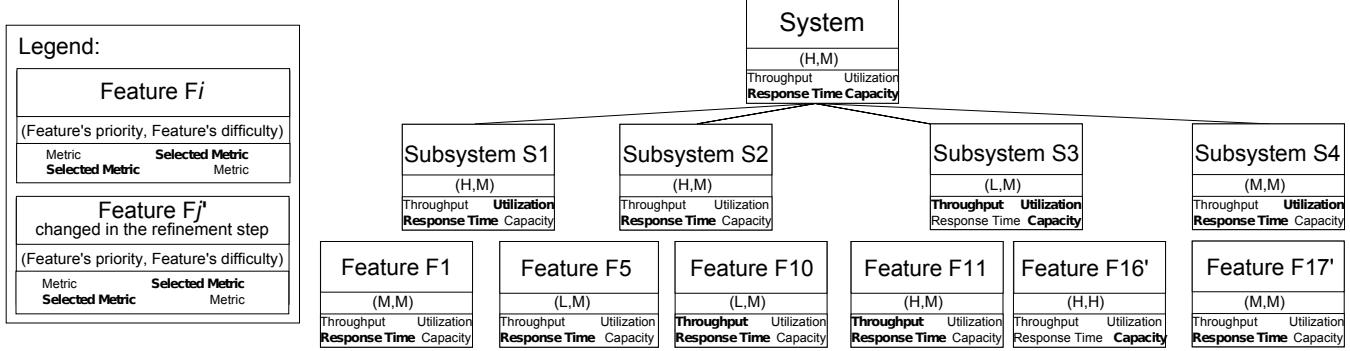


Fig. 4. The basic parts of our obfuscated feature model. Selected metrics are shown in bold font.

D. Requirements Validation

Throughout the process, we refined the preliminary results together with the stakeholders and sent status updates at least once per week. We found it important to keep them continuously involved. We often improved our results based on their feedback.

After having specified the requirements, we set up separate meetings with the development team, test team, and the management stakeholders to validate the requirements. We received much positive feedback, and these meetings also served to start the hand-over process to the development organization.

Since the requirements specification, the test team has used the requirements to create performance test cases. Initial tests have been successfully executed to set a performance baseline to which new versions of the DDSS can be compared. The test team did not discover any need for significant changes to the requirements.

VII. DISCUSSION

Through applying PROPRE in our case study, we learned its strengths and weaknesses. We discuss these in this section. Lessons learned not directly related to PROPRE are the subject of Section VIII.

Assumptions We expect that PROPRE generalizes to other systems, when its assumptions and constraints are considered:

- Only performance requirements need to be elicited.
- It may be troublesome to collect performance requirements at the system level or below the feature level, because of the feature model.

A. Strengths and Criteria Fulfillment

We set up a set of criteria to ensure that our requirements engineering method met our needs (Section IV) and compare the experiences with PROPRE in the following.

C1 Understandable: Our stakeholders appreciated the good overview of the DDSS's features in the feature model and found the technique easy to understand. It was sufficient to briefly present the exercise and clarify the metrics' definitions. One drawback that we encountered is the limited amount of

information a feature model gives when it comes to scenarios and actors for a feature. For our case study, this meant that we had to reconstruct a use case diagram to have sufficient information. The development team found the requirement specification templates very useful and easy to use. In most cases, they could directly fill in the information. One category that was difficult to specify was the *Condition of the System* section. It is difficult to foresee what the system workload will be at runtime, as it depends on, for example, the number of active users.

C2 Involved Stakeholders but C5 Distributed Environment: Stakeholders are directly included into the requirements engineering process. The PROPRE method is designed to be used in discussions and meetings. At the same time, The PROPRE method is suitable in a distributed business environment. Our method is flexible and does not require all stakeholders to be at one place at one time. In case stakeholders cannot participate in important meetings they can be contacted in later steps.

C3 Focus on Key Requirements: The focus on externally visible system features of the annotated feature model approach with our addition of a ranking step allowed to focus on key performance requirements only.

C4 Basis for Test Scenarios: The requirements specification step of the PROPRE method leads to performance requirements that form a good basis to specify test scenarios. It helps to precisely define the context of the performance requirements and facilitates the setup of performance tests. The specified requirements are now used to do structured performance testing of the DDSS.

C6 Applicable under Time Constraints: The feature model helped us to select only relevant use cases to reduce the time effort. We estimate that we spent two to three person weeks on the actual requirements engineering. The estimate excludes the time taken by our stakeholders. Given the size of the DDSS, the complexity of its organization (both described in Section II-B), and considering the lack of information, we find this a competitive time cost.

B. Weaknesses

Ranking Exercise Could Be Improved Prioritizing requirements and focusing on the most important ones is natural. Our stakeholders could discuss the rankings and metrics selection to a large extent without moderation. In some cases, it was necessary to revive or steer the discussion. For example, to ensure that all features were ranked.

The ranking method and dimensions we chose, however, were not clear to all of our stakeholders. We copied the difficulty ranking from the Attribute Driven Design method [11] to test it in another context, but conclude that its role is unclear. The difficulty ranking impacted neither the selection of requirements nor their specification. The inclusion of this ranking should be reconsidered.

It was difficult for our stakeholders to agree on which features should be ranked as Low. Since all features exist for a reason and are relevant for the system, the participants' tendency was to rank a lot of requirements as High.

We suggest to evaluate whether the ranking in High, Medium, and Low is optimal or if other scales and techniques work better. A ranking based on the Volere Shell such as *Customer Satisfaction* and *Dissatisfaction*, as used in our specification patterns, might be a more suitable approach.

Lack of Tool Support for Feature Models While not directly related to PROPRE, a problem is the lack of tools for feature models. Larger commercial tools such as Sparx Enterprise Architect support feature models, but do not offer free editors that stakeholders can use to annotate the model with their comments. Free tools often lack the ability to attach feature descriptions or scenarios to features in the model. As a consequence, an appropriate tool for feature-oriented performance requirements elicitation should be developed.

VIII. RECOMMENDATIONS

Working with a complex system in a corporate environment gives valuable insight into what practices are useful. In this section, we share recommendations that are not directly related to PROPRE. We hope that fellow and especially new practitioners find these useful.

Use Paper As Technology We have often used paper during meetings to communicate ideas and as a shared working space. Recall our feature sheet (Figure 3) and model prints (Section VI-B). We recommend others to try this.

Establish Common Terminology Partly because of the lack of available up-to-date documentation, we often had to rename features. The feature names were partly ambiguous and not specific enough, and feature models do not offer other visual clues or scenarios to define a common terminology. Next, importance of a feature may be difficult to understand, when not the presence of the feature but its quality has to be judged. A feature may be important to have, e.g., an administrator interface, but its quality can be less of a concern. Finally, stakeholders are unlikely to have the same definition of performance metrics, because not all are performance engineers or even

computer scientists. Therefore, we recommend to explicitly define terminology early on in the process.

Know Your Stakeholders and Their History Knowing who your stakeholders are and what their historical relationship to the system is enables you to ask the right questions to the right person. In our case study, for example, we decided to ask the development team about values for the metrics during specification, because unlike other teams it has worked on the system since its inception.

Nurture Stakeholder Relationships Whether you work in your own team or as an (internal) consultant, you have to gain the trust of your stakeholders and make them spend time with you. Therefore, you have to nurture your relationship with them. For us, frequently sharing status updates and results with *all* stakeholders helped. You may not get feedback, but, if well delivered, every message tells you care and want to help building a great system.

Plan to Iterate and Expect to Change Plans You will find that results of an earlier step could be improved, and it may be best to take a step back and revise. In practice, the waterfall model does not work for requirements engineering. Many things cannot be predicted. Cheesy examples are stakeholders on holidays or sick leave, but in reality it happens. Be prepared and adapt.

Proceeding iteratively is necessary when discovering requirements, but set clear goals on what to achieve. We set our goal at 20 detailed requirements within two months and our budget. This would enable the test team to test key cases and provide reasonable coverage.

IX. RELATED WORK

Several *methods* for handling quality requirements in the requirements engineering process have been suggested. However, most of these focus on specific phases of the requirements engineering process. We have included some of them into our overall performance requirements engineering process (cf. Section IV for elicitation methods and Section V for methods in other requirements engineering phases).

Most other methods for quality requirements we are aware of focus on the interactions of different quality attributes and thus are not relevant for handling performance requirements only. Examples are QUARCC [18] and NFD [19], which are methods to identify and resolve quality requirements conflicts.

There are several exceptions: QUPER [20] is a method for setting quality targets, which is an aspect of prioritizing quality requirements. As such, it could be integrated into PROPRE as an extension of the requirements specification step. Furthermore, Nixon presents the Performance Engineering Framework (PeRF) [21] based on the NFR framework. In contrast to our work, PeRF focuses on managing and fulfilling performance requirements by helping developers to operationalize them. The Quality of service Modeling Language (QML) [22] can be used to define quantifiable quality requirements, such as performance requirements. Thus, it could be used alternatively in PROPRE's requirements specification

phase. However, as the goal of PROPRE is to create human-understandable requirements, the natural language description in a template is probably more suited. The requirements process and Volere Shell by Robertson form a comprehensive requirements engineering tool [23]. Pointers and templates support the ‘discovery’ of various types of functional and non-functional requirements. We focus on performance requirements and extended the Volere Shell to get more specific performance requirements that are easier to test.

A large body of work has been devoted to quality models in the past, such as the ISO quality model [24]. Such quality models can be used as checklists when eliciting quality requirements for a project. In the context of this paper, however, the goal of specifying performance requirements is already set and we assume that the performance metrics of interest are also already known.

Furthermore, several *case studies* on requirements engineering methods for quality requirements have been presented. None of them, however, specifically focuses on performance requirements as PROPRE does.

Dörr et al. [5] have conducted three case studies adopting an NFR method. The method helps to create checklists so that non-functional requirements can be easily specified. Use case descriptions and a tailored quality model, that presents a high-level quality attribute (e.g., *Efficiency*) with lower-level attributes and metrics, form the basis to the creation of the checklists. Their method iterates over relevant functional elements and selects appropriate quality metrics for them. However, it relies on existing functional documentation which makes it difficult to deploy in our context [5, p. 7].

[25] presents a comparison between two approaches — Dörr’s method stated above and the MOQARE method which is a Misuse Case based approach. They applied both methods on the same system and describe their results and experiences. The study is in the context of security requirements.

X. CONCLUSION

In this paper, we have presented the PROPRE method which combines existing requirements engineering methods with a focus on practice. The method has been developed for use in industry, is easy to deploy, and overcomes several common assumptions. For example, it neither relies on a complete functional specification nor does it require all stakeholders to meet at the same time and place.

We have successfully applied the requirements engineering method to collect and specify the performance requirements of ABB’s DDSS. Deploying our method, we could deliver a set of precise performance requirements with two to three person weeks of effort. The requirements are now used to run regular performance tests according to a test plan.

We expect that the PROPRE method can be deployed in other contexts and maybe even for other quality attributes, but further case studies are needed to validate the method’s applicability in other situations.

ACKNOWLEDGMENTS

This work is supported by the German Research Foundation (DFG) within the Collaborative Research Centre “On-The-Fly Computing” (SFB 901).

REFERENCES

- [1] S. L. Pfleeger and J. M. Atlee, *Software Engineering: Theory and Practice*, 4th ed. Pearson, 2010.
- [2] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*. John Wiley, 1998.
- [3] U. Hammerschall and G. Beneken, *Software Requirements*, 1st ed. Pearson Studium, 2013.
- [4] A. B. Bondi, “Best practices for writing and managing performance requirements,” *Proc. of the 3rd Intl. Conf. on Performance Engineering (ICPE)*, pp. 1–8, 2012.
- [5] J. Dörr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, “Non-functional requirements in industry — three case studies adopting an experience-based NFR method,” in *Proc. of the 13th IEEE Intl. Conf. on Requirements Engineering*, 2005, pp. 373–382.
- [6] T. Wang, Y. Si, X. Xuan, X. Wang, X. Yang, S. Li, and A. J. Kavs, “A QoS ontology cooperated with feature models for non-functional requirements elicitation,” in *2nd Asia-Pacific Symposium on Internetworks*. New York, USA: ACM, 2010, pp. 17:1–17:4.
- [7] J. Dörr, “Elicitation of a complete set of non-functional requirements,” *PhD Theses in Experimental Software Engineering*, vol. 34, 2011.
- [8] S. Withall, *Software Requirements Patterns*. Redmond, Washington: Microsoft Press, 2007.
- [9] J. Mylopoulos, L. Chung, and B. Nixon, “Representing and using nonfunctional requirements: A process-oriented approach,” *IEEE Transactions on Software Engineering*, vol. 18, no. 6, pp. 483–497, Jun. 1992.
- [10] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Springer, 2000.
- [11] M. Barbacci, R. Ellison, A. Lattanzi, J. Stafford, C. Weinstock, and W. Wood, “Quality attribute workshops,” Architecture Tradeoff Analysis Initiative, Tech. Rep., 2003.
- [12] A. Herrmann and B. Paech, “MOQARE: misuse-oriented quality requirements engineering,” *Requirements Engineering*, vol. 13, no. 1, pp. 73–86, Sep. 2007.
- [13] R. Wohlrab, “Elicitation methods for performance requirements: Requirements engineering in industry,” Bachelor’s Thesis, University of Paderborn, 2013.
- [14] L. Chung and B. Nixon, “Dealing with non-functional requirements: three experimental studies of a process-oriented approach,” *17th International Conference on Software Engineering*, pp. 25–37, 1995.
- [15] J. Robertson and S. Robertson, *Mastering the Requirements Process*. New York, USA: ACM Press/Addison-Wesley Publishing Co., 1999.
- [16] L. Westfall, *Software Requirements Engineering: What, Why, Who, When, and How*. The Westfall Team, 2005.
- [17] R. Wójcik, F. Bachmann, L. Bass, and P. Clements, *Attribute-Driven Design*. Software Architecture Technology Initiative, 2006.
- [18] B. W. Boehm and H. In, “Identifying quality-requirement conflicts,” *IEEE Software*, vol. 13, no. 2, pp. 25–35, 1996.
- [19] E. R. Poort and P. H. N. de With, “Resolving requirement conflicts through non-functional decomposition,” in *4th Working Conf. on Software Architecture (WICSA)*, 2004, pp. 145–154.
- [20] R. Berntsson Svensson, Y. Sprockel, B. Regnell, and S. Brinkkemper, “Setting quality targets for coming releases with QUPER: an industrial case study,” *Requirements Engineering*, vol. 17, no. 4, pp. 283–298, 2012.
- [21] B. Nixon, “Management of performance requirements for information systems,” *IEEE Transactions on Software Engineering*, vol. 26, no. 12, pp. 1122–1146, 2000.
- [22] S. Frølund and J. Koistinen, “Quality-of-service specification in distributed object systems,” Hewlett Packard, Tech. Rep., 1998.
- [23] S. Robertson and J. Robertson, *Mastering the Requirements Process*, 3rd ed. Upper Saddle River, NJ, USA: Pearson Education Inc., 2012.
- [24] ISO/IEC 9126-1:2001(E), *Software engineering – Product quality – Part 1: Quality model*. International Organization for Standardization, Geneva, Switzerland, 2001.
- [25] A. Herrmann, D. Kerkow, and J. Dörr, “Exploring the characteristics of NFR methods: a dialogue about two approaches,” *Requirements Engineering: Foundation for Software Quality*, pp. 320–334, 2007.

Lightweight Requirements Engineering Assessments in Software Projects

Daniel Rapp¹, Anne Hess², Norbert Seyff³, Peter Spörri¹, Emmerich Fuchs¹, Martin Glinz³

¹ Zühlke Management Consultants AG
Schlieren, Switzerland
{dra, psp, efu}@zuehlke.com

² Fraunhofer IESE
Kaiserslautern, Germany
anne.hess@iese.fraunhofer.de

³ University of Zurich
Department of Informatics
Zurich, Switzerland
{seyff, glinz}@ifi.uzh.ch

Abstract—Requirements engineering (RE) is widely recognized as a crucial factor for the success of software projects. Therefore, companies often request assessments of RE processes and resulting artifacts to identify issues and improvement potential. However, industry claims that current assessment approaches do not always fulfill their needs regarding efficiency and effectiveness. Motivated by needs of both, companies asking for an assessment, and a company in the role of an assessor, we have developed a lightweight, tool-supported RE assessment approach. Apart from presenting the approach, we also discuss early experiences we gained from applying our assessment approach in real-world industrial projects.

Index Terms—Requirements engineering assessment, requirements process improvement

I. INTRODUCTION

It is widely recognized in industry that successful requirements engineering (RE) has a tremendous effect on the success of software engineering projects [1] [2]. Hence, when a company becomes aware of problems with its RE processes, it should perform an *assessment* of these processes and then, based on the assessment results, take action for improvement.

Nevertheless, we observe situations where companies experiencing troubles with their RE processes do not take action for improvement or do not act effectively and efficiently. A major reason for such behavior is that companies do not know how to assess their RE processes properly or deem such an assessment to be too complicated and expensive. Other reasons, which we do not address in this paper, could be that such companies lack defined software processes, do not have defined process improvement procedures, are short of resources for process improvement, or just do not recognize the need to change their RE practices [3] [4].

Identifying and choosing an assessment method for RE processes that is effective and efficient in a given situation is indeed a challenge [5] [6] [7].

Several assessment ideas and also concrete concepts including questions have been published in academia and are publicly available [8] [9] [10]. Since these ideas are distributed over different publications, they are hard to identify for companies. Furthermore, these approaches often do not provide clear guidance regarding their actual application and the

analysis of results. Likewise, suitable tool support is often limited.

On the other end of the spectrum there are approaches such as CMMI [11] and SPICE [12] which can be applied to a broad range of companies and which assure a standardized assessment. Typically, these assessments allow a company to get professional support by assessors and receive a “rating”, which can be communicated to their customers and be used for comparisons to other companies. However, often little details about the assessment method are communicated to companies upfront. This can leave them in uncertainty on what to expect.

For example, concrete questions are often not available to the public which makes it hard to understand to what extent and how a particular assessment approach addresses RE-related problems. Moreover, the application of such approaches requires a certified assessor with specific knowledge. The high cost for such an assessor can be a major obstacle for companies. Finally, preparing for and conducting CMMI or SPICE assessments is time-consuming.

These facts have motivated us to develop the *RE Assessment Guide*, a lightweight, tool-supported RE assessment approach which provides industrial companies with the opportunity to assess their current RE processes efficiently and effectively. The RE Assessment Guide allows to answer questions about the quality levels of RE processes applied in a software development project or organization, such as:

- “Is there a defined RE process to be followed during a particular project?”
- “What kind of activities are performed within the RE process and how are these activities performed in detail?”
- “How are the outcomes of the RE activities documented?”
- “What is the quality of the documented information?”

In our experience, industry requests professional support to answer those questions. So in order to achieve reliable and meaningful assessment results, they want experts to conduct such assessments. Therefore, the assessments conducted with the RE Assessment Guide include an RE expert in the role of the assessor.

The remainder of the paper is organized as follows. Section II highlights needs regarding efficient and effective

assessments. In Section III we discuss existing assessment strategies that have been investigated during the development of the RE Assessment Guide. This discussion is followed by the introduction of the RE Assessment Guide in Section IV. Section V presents experience we gained from first assessments conducted with the help of the RE Assessment Guide in real software projects. The paper concludes with a short summary and outlook on future work in Section VI.

II. INDUSTRY NEEDS REGARDING RE ASSESSMENTS

The RE Assessment Guide is motivated and driven by needs of companies asking for an assessment on the one hand, and companies in the role of assessors on the other hand. We have gathered concrete industry demands regarding RE assessments in discussions with customers over several years. Additionally, we have elicited assessor needs internally in discussions with senior RE consultants.

The subsequent presentation of the elicited needs is structured according to the two viewpoints mentioned above. We distinguish between needs regarding the preparation, the execution, and the analysis phase of an assessment within an organization.

A. Viewpoint of Companies in the Role of Customers

From the viewpoint of companies requesting an RE assessment, the *preparation phase should not require additional effort and should not result in long waiting times until the actual assessment can be conducted* (C1). The execution of the assessments should be efficient, i.e., *the execution should neither involve too many resources, nor require too much time or generate too high costs* (C2). Furthermore, *assessment questions should be easy to understand and to answer* (C3). The analysis phase should be effective, i.e., *the analysis should deliver meaningful, comprehensible, and (statistically) reliable results tailored to particular company needs* (C4). Equally important is that *results should be objective and repeatable, i.e., if more than one expert independently assesses the same process, the analysis should deliver similar results* (C5). *Besides weaknesses and improvement potential, the analysis should also reveal current strengths which should be further retained in the future* (C6). Apart from that, *companies should be provided with clear guidance on how to overcome identified weaknesses* (C7). Finally, *the analysis should also reveal a "rating" illustrating the current standing of the company in comparison to other companies* (C8).

B. Viewpoint of Companies in the Role of Assessors

From the viewpoint of companies which would like to conduct assessments, an efficient and effective preparation should be supported. That is, *it should be easy to adapt the assessment to specific company and project characteristics and to particular customer needs* (A1). Furthermore, *the preparation phase should not take too much time and involve too many stakeholders from the customers' side to deliver information* (A2). For members of a company offering such assessments, *it should be easy to learn how to conduct the assessments* (A3). *No exhaustive and expensive training should*

be necessary, assuming that future assessors already have RE knowledge (A4).

The execution of an assessment should also be efficient and effective. That means that *the execution should not require too much time, generate too high costs and require too many resources* (A5). Apart from that, *the assessor should be able to tailor and focus assessment questions to particular information gained during an assessment session and detect connections between different answers* (A6). In order to finally deliver meaningful results, *it should also be possible to get deeper insight into interesting aspects during discussion and to easily document discussion results* (A7).

Finally, there are also several demands regarding efficiency and effectiveness of the analysis phase where information collected during an assessment is analyzed and communicated to the customer. These demands include *fast and reliable result analysis and preparation (e.g. by using statistical analysis approaches)* (A8), *being able to get a good understanding of current problems, their reasons and their severity to identify suitable improvement strategies* (A9).

III. RELATED WORK

This section gives an overview on existing assessment strategies that have been investigated during the development of the RE Assessment Guide. We briefly introduce each approach and discuss it with respect to its relation to the RE Assessment Guide.

The *Capability Maturity Model Integration* (CMMI) [11] and the *Software Process Improvement Capability Determination* (SPICE) [12], also known as ISO/IEC 15504, are prominent maturity models for assessing and improving software development processes [13]. In order to determine the maturity level of a particular software development process, key software engineering activities are analyzed, including RE activities. The latter activities comprise, for instance, *develop customer requirements, develop product requirements or analyze and validate requirements*. Since the application of these models is quite extensive, their usage for conducting RE assessments is rather complex and time consuming [5]. Thus CMMI as well as SPICE are inadequate for performing an efficient RE process assessment within a short amount of time. However, both approaches give a good overview on how RE activities have to be performed and have served as valuable input during the development of the RE Assessment Guide.

The *Software Product Management Maturity Matrix* (SPMMM) [8] provides guidelines on how to optimize software product management processes. To determine the maturity level of a software product management process, requirements management activities and related tasks are also investigated. The detailed descriptions of these activities and tasks have been considered as a basis for the assessment criteria of the RE Assessment Guide. This input includes especially tasks related to activities such as *requirements gathering, requirements identification, requirements organizing, requirements prioritization, release definition and validation, and scope change management*. However, SPMMM is not RE-specific, but focuses on the entire software product

management lifecycle. For example, the SPMM does not consider requirements artifacts in detail. Hence, this framework cannot be used out-of-the-box for assessing the quality of a given RE process.

Within the scope of the *ReqMan* research project, a framework has been developed which aims at optimizing RE processes of small to medium-sized enterprises (SME) [4] [14]. *ReqMan* investigates typical RE-related activities such as *requirements elicitation*, *requirements analysis*, *requirements specification*, *requirements verification and validation* as well as *requirements management*. For each of these activities, the framework defines basic, advanced and optimized practices and related techniques. Considering, for example, the activity *requirements validation*, a basic practice would be *review requirements* with a corresponding technique like *checklist-based reading*. An advanced practice for this activity would be *prepare tests for requirements*, and an optimized practice would be *prototyping* [14]. Even though the aim of *ReqMan* is quite similar to the aim of the RE Assessment Guide, i.e., allow efficient and effective RE assessments, the *ReqMan* solution approach is different. *ReqMan* provides a tool-supported method that enables companies to evaluate and improve their established RE processes on their own without the necessity to hire an external assessor. That is, based on the detailed descriptions of activities, practices and related techniques that are provided by the framework, companies can easily evaluate their current implementation status of RE practices and identify and incorporate improvement potential. Similar to the activities and tasks defined within the SPMM, the detailed descriptions of practices have been used as input for assessment criteria of the RE Assessment Guide.

The *Requirements Capability Maturity Model* (R-CMM) [9] describes what kind of RE processes an organization has to establish to reach a certain maturity. R-CMM is based on the RE processes defined in CMMI, and has the intention to specify the rather abstract CMMI RE processes in more detail. The aim of R-CMM is to provide a framework for companies that supports the implementation of their RE processes according to a target process maturity. Since processes and sub-processes are solely defined for maturity level 2 (repeatable software processes), R-CMM does not suffice to conduct an entire assessment and to determine the maturity level of the RE process in a concrete software development project. However, the detailed process definitions regarding maturity level 2 have been implemented as assessment criteria in the RE Assessment Guide.

The *Requirements Engineering Reference Model* (REM) [10] is a reference model which defines RE artifacts, their desired contents and their dependencies. In their work, Geisberger et al. describe in detail which artifacts are needed within an RE process and what the aim of each of these artifacts is. Especially the part *requirements specification* has turned out to be a relevant source for the RE Assessment Guide. We have used it as a basis for defining the assessment criteria for the document analysis part. However, due to its specialization on RE artifacts, REM is not appropriate to be

used solely for the execution of assessments regarding the entire RE process in software development projects.

To summarize, there are several maturity models and assessment approaches that support the analysis of current RE processes established in software development companies. These approaches include highly useful concepts and each approach supports a specific purpose. Some of these ideas have been incorporated in our RE Assessment Guide. However, having conducted this analysis, we also concluded that none of these approaches fulfills our needs regarding efficient and effective RE process assessments as discussed in Section II. For example, some approaches focus on the complete software engineering process and investigate RE-related activities only on a quite abstract level, as it is the case for SPMM. Other approaches do aim at enabling efficient and effective RE process assessments, but follow a different solution approach such as cooperative self-assessments provided by the *ReqMan* framework. Other approaches focus on particular assessment aspects only. For example REM concentrates on the quality of RE artifacts and does not consider the RE process or activities in detail.

As a result, we decided to develop an independent RE Assessment Guide on the foundation of the strengths of the described maturity models and assessment frameworks.

IV. THE RE ASSESSMENT GUIDE

The development of the RE Assessment Guide was motivated by our vision of a lightweight and structured method to answer questions regarding the quality of RE processes (see Section I) and to meet demands of both industry and assessors (see Section II).

In addition, it should be possible to adapt the RE Assessment Guide to the characteristics of a concrete software project and the developing organization to consider the heterogeneity of different software development projects. Furthermore, to support a standardized assessment, the RE Assessment Guide should be implemented in a tool to be used by the assessor which also supports the result analysis of the conducted assessments.

We have developed the RE Assessment Guide in an iterative manner, testing early prototypes within internal projects and using the feedback gathered for guiding the further development. The assessment criteria that we have implemented in the RE Assessment Guide mainly stem from an analysis of existing assessment approaches (as introduced in Section III) as well as from RE related literature such as [15] [16] [17] [18] [19]. We have identified further assessment criteria by means of expert interviews. We have conducted these interviews to get an impression of how RE is performed in practice within software development companies.

In the remainder of this section, we describe the implementation structure and application of the RE Assessment Guide in detail.

A. Implementation Structure

Technically, the Assessment Guide is implemented in LimeSurvey [20], an open-source-tool for creating and

conducting surveys. In its current version [22], the RE Assessment Guide is divided into three main parts: (1) influencing factors, (2) process analysis, and (3) document analysis. While part 1 is always executed, it is possible to decide whether the assessment should only include document analysis, process analysis or both. For example, if a customer is primarily interested in assessing the quality of artifacts created during current RE activities, only the document analysis part can be conducted together with the analysis of influencing factors. The assessment criteria of each of these parts are implemented in the form of concrete questions which can be answered on a scale from 1 = “Definitely yes” to 4 = “Not at all” and be supplemented with comments. There is also the option to give no answer to a particular question. Figure 1 shows an example question from the RE Assessment Guide with the five answer options and a field for free-form comments. The assessor fills in this survey during an assessment interview based on the answers given by the interviewee (see step 3 in Section IV.B) or answers these questions during the document analysis (see step 4 in Section IV.B).

Fig. 1. RE Assessment Guide – Example Question

Subsequently, we describe the three parts and corresponding assessment criteria in more detail.

Part 1 - Influencing Factors

This part includes assessment criteria related to general project-specific information, comprising, for instance, company size, business areas, and roles that are involved in an investigated project. Assessment criteria of this part include questions such as:

- “In what kind of business areas is your company operating?”
- “How many employees does your company employ?”

- “How many full-time equivalents are working in the investigated project?”
- “What roles do these equivalents have within the investigated project?”

Part 2 - Process Analysis

In this part, detailed information is elicited regarding underlying process models (e.g. waterfall approach or iterative approach), RE processes and activities followed as well as communication flows within the project. Assessment criteria related to process analysis include:

- “Do you follow a defined RE process?”
- “Do you perform the following RE activities, e.g. elicitation, validation, negotiation, documentation and management?”
- “Do you integrate all relevant stakeholders in requirements elicitation, e.g. customers, internal stakeholder, partners or end-users?”
- “Do you prioritize your requirements together with your stakeholders?”
- “Do you have a defined requirements change management process in place?”

Part 3 - Document Analysis

This part is dedicated to eliciting detailed information about artifacts / documents that have been specified within an investigated project. For each artifact, detailed information is captured regarding notations, when it was initially created (e.g. in the inception phase), if and when it was refined (e.g. during elaboration time), when it was communicated to other team members, etc. Further assessment criteria regarding the document analysis part include, for example:

- “Do you iteratively create your documentation?”
- “Is there a unique identifier for each requirement?”
- “Do you document the status for each requirement?”
- “Do you document the author for each requirement?”
- “Do you document the acceptance criteria for each requirement?”
- “Do you link dependent requirements?”
- “Do you link requirements with test cases?”

B. Assessment Lifecycle with the RE Assessment Guide

The RE Assessment Guide focuses on project rather than company assessments. However, identifying “typical” projects or assessing a larger number of projects within a company is supported and allows to draw conclusions about the general quality of the RE process within a company. Typically, a project assessment is conducted by an assessor who interviews one or more representatives of the customer company who have been involved in the RE activities of the project under investigation. The RE Assessment Guide guides the assessor through the various questions eliciting information about

influencing factors, processes and documents, whereby it dynamically adapts the questions to given answers based on predefined dependencies. For example, only if the question “do you integrate the customers in the requirements elicitation?” is answered with “definitely yes” or “rather yes”, the subsequent question “How often do you elicit requirements from the customer?” is displayed in the RE Assessment Guide. Similarly, if there are no use cases created within the project, the assessment criteria regarding how the use cases are documented in detail will not be asked.

As illustrated in Figure 2, a typical assessment lifecycle with the RE Assessment Guide comprises six steps that we describe subsequently. The information about participants and duration of each step are based on our first experience with conducting assessments (see Section V.A).

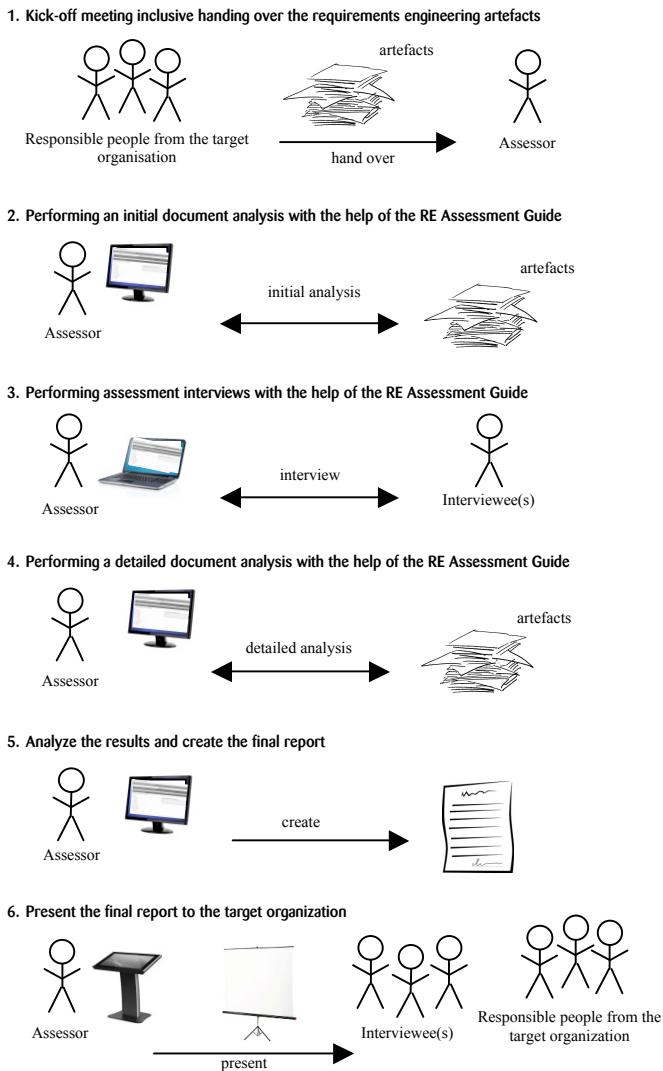


Fig. 2. Assessment Lifecycle with the RE Assessment Guide

Step 1 - Kick-off Meeting

(Participants: assessor and customer representatives; Duration: ~1 hour)

In the first step of an assessment, the assessor organizes a kick-off meeting with the customer, i.e., the company that requests the assessment. The goal of this meeting is to explain how an assessment is conducted and to discuss organizational issues such as the time plan for the interview session(s) (see step 3) and the final presentation (see step 6). Shortly after the kick-off meeting, the customer hands over relevant RE artifacts, (e.g. the requirements specification) which will be analyzed by the assessor in the subsequent steps 2 and 4.

Step 2 – Initial Document Analysis

(Participant: 1 assessor; Duration: ~4 hours)

In this step, the assessor initially analyzes the RE artifacts that were handed out by the customer (see step 1). The goal of this analysis is to get a first impression about how RE is done at the customer’s side. Additionally, this analysis serves as a means for initial identification of interesting aspects and open issues that should be focused on and discussed in further detail during the interview session (see step 3). The output of this step is an overview of existing artifacts and questions about the artifacts which can be raised during the interviews.

Step 3 – Assessment Interviews

(Participants: 1-2 assessors and 1-5 customer project team members; Duration: ~2 hours per interview)

The goal of this step is to perform an assessment interview with one or more interviewee(s) who are actively involved as team members in the investigated software development project. In order to get a differentiated view on the RE process, it is possible to involve interviewee(s) who have different roles within the software development project. This step results in detailed insight into how the RE process is performed within the investigated project. Optionally, follow-up interviews may be conducted to clarify any issues raised in interviews or found after interviews, for example, by cross-checking initial interview results and documentation.

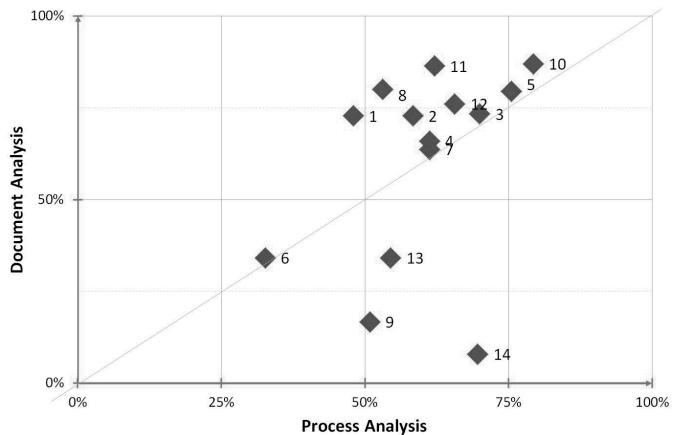


Fig. 3. Matrix visualizing comparisons of different assessments. The matrix shows the percentages of questions in the parts ‘Process Analysis’ and ‘Document Analysis’ that were answered with “definitely yes” or “rather yes”. Each diamond represents one of the conducted assessments.

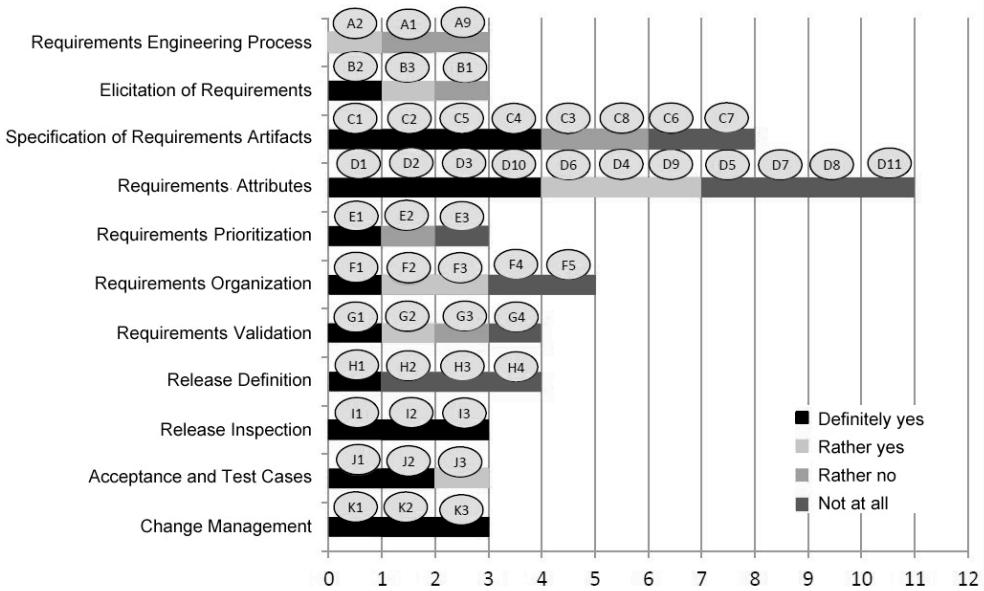


Fig. 4. Extract from result analysis visualizing results of all assessment criteria (A2 – K3) grouped by the assessment categories (Requirements Engineering Process, Elicitation of Requirements, Specification of Requirements Artifacts, etc.)

Step 4 – Document Analysis Continued (Participant: 1 assessor; Duration: ~8 hours)

Similar to step 2, a document analysis is performed with the help of the RE Assessment Guide. This time, it is a more detailed analysis on the basis of the insights gained from the assessment interview(s) conducted in step 3. At the end of this document analysis activity, the investigation of the artifacts is finished and all assessment criteria included in the assessment guide are answered.

Step 5 – Result Analysis and Report Creation

(Participant: 1 assessor; Duration: ~16 hours)

After conducting the interview(s), the assessor has to analyze and evaluate the captured data both from the document analysis (steps 2 and 4) as well as from the conducted interview(s) (step 3). Result analysis is supported by diagrams which can be generated with the LimeSurvey tool [20] to get an overview about the quality of the assessed RE process. Figures 3 and 4 show examples of result diagrams.

All results gained by the data analysis are documented in a final report. The report summarizes the major findings of the quality assessment of the investigated RE process, including strengths, weaknesses and improvement potential. Thus, the outcome of step 5 is the final report which will be handed over to the customer.

Step 6 – Final Presentation

(Participants: 1 assessor and customer representatives; Duration: ~1 hour)

Finally, the last step of the entire assessment lifecycle is to present the results of the RE assessment to the customer in a meeting. Relevant people from the customer's side who have an interest in the assessment results (such as team members of

the investigated software development project, members of other projects or the management of the company) should be invited to this meeting.

Analysis of Overall Effort

In summary, conducting a typical assessment with the RE Assessment Guide requires the following overall efforts both from customer's and assessor's sides:

- Each customer representative participating in the kick-off-meeting (step 1), the interview (step 3) and the final presentation (step 6) spends about four hours overall. In a typical project setting, there are three team members involved. Thus, based on our experience, the overall effort on the customer side is about twelve hours. Of course, the effort required depends on the organization's setup for the assessment. For example, the effort will increase if the organization decides to invite a larger audience to the final presentation meeting.
- One assessor is required for the kick-off meeting (step 1), document analysis (steps 2 and 4), report creation (step 5) and final presentation (step 6). We often had two assessors who were conducting the assessment interviews (step 3). Typically, we conducted three assessment interviews. This results in a total of about 42 hours spent on the assessors' side.

V. FIRST EXPERIENCE

In this section, we present early experience we gained from the application of the RE Assessment Guide in real-world projects. The lessons learned discussed in this section summarize our own experience gained by different assessors (three of the authors) who have used the RE Assessment Guide

in customer projects. Furthermore, the lessons also reflect customer feedback we gained during discussions with customer representatives who have been actively involved in the project assessments. We first introduce the project settings and particular goals. Then we report on the lessons learned.

A. Assessment Project Settings and Goals

So far, the RE Assessment Guide has been applied in ten industrial software development projects. These projects typically involved three to seven full time equivalents, comprised a total effort of 600-1500 person days, and had a project duration of 12-18 months. Nine out of the ten projects followed an iterative software development process, whereas one project followed a waterfall approach.

The ten assessment projects can be categorized into three types of assessments based on the individual customer needs.

- 1) The goal of three assessment projects was to analyze the quality of current RE processes. The assessment was conducted after the elaboration phase (following the Rational Unified Process [21] in order to identify possible weaknesses and improvement potential that could be addressed in later phases (i.e. the construction phase).
- 2) The second assessment type was targeted to compare how RE is performed in several software development projects within one company. This assessment can, for example, be used to evaluate how a defined RE methodology is applied in different project settings. This assessment was conducted for six projects in two companies (two projects in one company, and four projects in the other company).
- 3) The third type of assessment conducted in one company had its focus on the investigation of communication flows within the project teams. That is, the assessment interview was conducted with different roles that were involved in the investigated projects. The goal was to investigate how requirements are communicated (e.g., in meetings, in different documents or in centralized documents) and to reveal any problems (e.g., whether the documentation of requirements is sufficient for testers to derive test cases, or whether relevant requirements artifacts are missing from the viewpoint of software architects, etc.).

Besides the projects mentioned above, the assessment guide was also applied within 27 projects in a semi-industrial environment. This means that up to 9 undergraduate computer science students in their first and second year were acting as a software development team and had a real industrial customer. To our experience, these projects are very similar to software development projects in industry, as a “real-life” customer provides a problem, which needs to be solved by a software system. Thus the project team has to deliver running software by a certain deadline just as real-world software development projects have to. Moreover, the project teams had to perform the entire software development process from the first contact with the customer to the delivery of the software system on their own. The project team members had to communicate directly and independently with the customer, identify relevant stakeholders and further information sources, and decide which artifacts to produce at what point of time in the project.

In this context we conducted assessments with the help of the RE Assessment Guide to evaluate the quality of the RE processes applied in early project phases (i.e., during the elaboration phase based on the Rational Unified Process [21]). The goal was to identify weaknesses which should be improved to avoid problems in later project phases (similar to the first type of assessments mentioned above). We are aware that these student projects are not fully equivalent to industrial projects. However, these projects also allowed us to apply and test the RE Assessment Guide in a homogenous, stable and comparable environment that we could control to some degree.

B. Lessons Learned

In this sub-section, we discuss the lessons learned from applying the RE Assessment Guide in real-world projects. The lessons are primarily based on the experience gained during the application of the RE Assessment Guide in the ten industrial software development projects. However, some lessons (especially those from the assessor’s viewpoint) have also been experienced in the semi-industrial setting. The lessons are partially traced to the industry needs discussed in Section II of this paper.

The flexibility of the RE Assessment Guide enables tailoring the assessment to customer-specific projects and assessment needs (A1, A6). As discussed previously in Section V.A, the RE Assessment Guide has successfully proven to be applicable in different project settings with different customer goals. Furthermore, it is possible to use the RE Assessment Guide in various settings, such as single projects, programs which comprise several projects, business unit methodologies, competence centers which are responsible for RE in a company, etc.

The RE Assessment Guide can be applied with a limited amount of time and resources (A2, A5, C1, C2). We found that companies clearly do not want to spend much time on assessments. However, a typical assessment lifecycle (as discussed in Section IV.B) requires customer involvement during certain steps in the assessment. The overall effort that the customer has to spend during a particular project is about four hours per involved person (see also Section IV.B) based on our experience gained during the assessments conducted so far. None of the customers complained about the time needed for the assessments. On the contrary, some customers were even positively surprised about the required effort. They initially expected to have to spend more time in the assessment process. However, the limited amount of time and resources needed from the companies needs to be compensated by an in-depth document analysis which is done solely by the assessor (see steps 2 and 4 in Section IV.B). This analysis delivers meaningful results to the customer without requiring any time resources on the customer’s side.

High quality specifications reduce the time involvement needed from customers (C2). We have experienced that the quality attributes of a specification (e.g., completeness, traceability, understandability) also influence the time needed for an assessment. In particular, high quality specifications enable the assessor to perform a thorough document analysis (see steps 2 to 4 in Section IV.B) without the need of extensive

interaction and communication with customer representatives. However, in case of low quality specifications, missing information needs to be discussed with customer representatives and more interviews are necessary in order to identify reasons and get explanations.

Assessment questions are easy to understand by the customers (C3). The application of the RE Assessment Guide has revealed that the assessment questions of the current version of the RE Assessment Guide are easy to understand by the interviewees, so that they can give suitable answers to these questions. Applications of previous prototype versions of the RE Assessment Guide have revealed some inconsistencies and ambiguities in the questions which have been tackled during the iterative development of the RE Assessment Guide.

Comments and open discussions are very important (A7, A9). As described above, the standardization of answers to the assessment questions is very useful to allow comparisons between different assessments. These answers alone, however, are not sufficient to identify suitable improvements and to provide guidance to the customer to realize these improvements. Therefore, it is very important to collect and record any comments given by the customers, which basically provide the rationale for certain answers.

Selection of suitable interviewees and moderation skills by the assessor are very important. We found that it is important to include the “right” people to get meaningful results. In order to get true answers, it is also very important that the assessment does not have the smell of marking the customer, but to find possible improvements.

Differences in answers given by several people are hard to detect during the assessment interview (A6). The possibility to conduct the assessment interview with up to 5 persons (even with different roles in a particular project) is also considered to be very valuable for customers as it enables to share their opinions regarding RE activities, which in turn fosters the communication between the team members. During the application of the RE Assessment Guide with several people involved in one particular project (see third project type in Section V.A) we experienced that different people have sometimes different opinions regarding a particular assessment criterion. For example, one interviewee in the role of a requirements engineer totally agreed that all requirements are accessible at a centralized location whereas a tester claimed that this criterion is not fulfilled at all. In such cases it would be very interesting to further discuss such diverging opinions, but this requires that the assessor is aware of them during an assessment interview. However, this is quite hard especially when several assessments interviews are conducted within one project.

Document analysis is very beneficial. Our experience highlights that the document analysis activities (see step 2 and step 4 in Section IV.B) deliver meaningful results. That is, the document analysis which is conducted prior to the assessment helps to get initial insights into current RE activities and to identify interesting aspects that should be further discussed during the assessment interview(s). The second document analysis, which is conducted after the assessment interviews

have been performed, focuses on the detailed analysis of the documents with the help of the RE Assessment Guide. These results can then be included in the final report to support and supplement any descriptions of findings revealed during the data analysis of the interview result. Even though the document analysis requires additional effort for the assessors and is often not so easy, it helps to improve the quality of the results delivered to the customer and hence to increase the customer satisfaction.

The assessment delivers meaningful results to the customers (C4, C6, C7, C8). Meaningful and understandable results are very important outcomes for the customers. This includes feasible and manageable suggestions regarding possible improvements supplemented with concrete actions to realize these improvements. Based on the customer feedback we gained from the initial assessments during discussions, we conclude that the assessment results are very helpful to the customers to detect weaknesses and improvement potential within their current RE processes. Especially the fact that answers to the assessment questions are standardized (i.e., answers can be given on a scale from “definitely yes” to “not at all”, cf. Section IV.A) makes it possible to compare assessment results of a particular project to other conducted assessments (see Figure 3). Thus, it is possible to compare different projects within one company, but also compare the current standing of a company in comparison to other companies. Especially the latter aspect is worthwhile for the customers (see requirement C8 in Section II.A). However, in order to improve such comparisons of assessment results with other similar companies or projects, *it would be necessary to provide more statistical data*, e.g. how projects with a similar setup in terms of full time equivalents, budget and stakeholders have been assessed or which artifacts have usually been created in comparable projects. Even though it is currently possible to gather this data during each assessment, a detailed analysis and preparation of such comparisons is not efficient yet and hence the corresponding requirement (C8) is not fully met. This is mainly due to the fact that the LimeSurvey analysis engine is not very powerful. So huge manual effort is required for creating valid statistical data analyses. Future work aims at improving this issue.

The assessment delivers objective and repeatable results to the customers (C5). The presented RE Assessment Guide is a framework, which was designed to support RE experts in conducting RE assessments. The questions guide the expert, but are presented at a level of detail where it is still up to the expert to make interpretations and to tailor the questions to the actual project under analysis (e.g., the interpretation of the question “Do you integrate all relevant stakeholders in requirements elicitation” (see Section IV.A), could vary from one expert to another due to different opinions regarding relevance of stakeholders). This could mean that different experts would come up with different results using the RE Assessment Guide in the same project. However, this is not what we experienced in the assessments conducted so far. For some projects, more than one expert was involved and both experts performed the document analysis individually. In those

cases there was hardly any difference in the outcomes. In our opinion, this can be explained by the fact that both experts share a similar understanding of RE, which should be the case for experts within one company.

Report generation requires too much effort (A8). The main effort for the assessors is the creation of the final report. As described in Section IV.B, this activity currently requires about sixteen hours of work, especially since the tool chain (LimeSurvey, office tools) is not fully automated. In the current version of the tool, a semi-automatic report generation based on Visual Basic Macros is already possible. However, there is still a lot of work which has to be done manually such as copying the diagrams into the final report or maintaining the results matrix. As soon as it is possible to generate the structure of the final report including diagrams and comparisons to other projects automatically, the duration of one assessment could be further reduced.

Refinement of possible answers would be helpful. To allow comparisons between different assessments, the assessment criteria of the RE Assessment Guide have been implemented in the form of questions that can be answered on a scale from 1 = “Definitely yes” to 4 = “Not at all” (cf. Section IV.A). However, the answers that are given are currently based on the personal evaluation of the interviewee. Often it is also hard to distinguish between “Definitely yes” and “Rather yes”. To achieve more reliable results, it would be helpful to supplement the possible answers with further criteria that provide guidance to select an appropriate rating. For example the answers of the assessment criteria “Are the requirements prioritized?” can be refined into “Definitely yes: at least 90% of the requirements are prioritized”, “Rather yes: 60 – 90% of the requirements are prioritized”, “Rather no: 30 – 60% of the requirements are prioritized” and “Not at all: <30% of the requirements are prioritized”.

The RE Assessment Guide requires maintenance effort. At the moment it is hardly possible to add or remove assessment criteria and their relations while keeping collected assessment data for comparisons. Such updates require changes in the database, i.e., a new database has to be set up and all assessment data which has already been gathered has to be copied to the new database. This effort increases with the number of conducted assessments.

The RE Assessment Guide can be used without extensive training, but its application requires RE knowledge (A3, A4). The first applications of the RE Assessment Guide by different people have revealed that the guide and tool can be easily used without extensive training. However, in order to lead discussions during the interview and to analyze the results regarding strengths, weaknesses and improvement potential, the assessor needs to have sound RE knowledge. We have also observed that after the RE Assessment Guide has been used several times, the assessors become familiar with the questions so that the interview can be conducted freely without “sticking” to and focusing on the tool.

The RE Assessment Guide supports providing “ad-hoc” RE assessments. As the RE Assessment Guide is easy to understand and learn by a person who has RE knowledge, a

broader range of potential assessors can be made available in a relatively short period of time. Having an adequate number of assessors also means that waiting times for companies can be quite short if needed. Companies can further select an assessor who suits their needs best. For example, an assessor who has some domain knowledge or is locally available can be chosen.

VI. SUMMARY AND FUTURE WORK

Based on a discussion of industry needs regarding RE assessments both from the viewpoint of companies requesting RE assessments, and of companies in the role of an assessor, the paper has introduced the RE Assessment Guide – a lightweight tool-supported RE assessment approach. Besides a discussion of related work, the paper has described the implementation structure as well as a typical assessment lifecycle with the RE Assessment Guide.

Finally, the paper has shared early experience that we gained during 37 project assessments that have successfully been conducted with the help of the RE Assessment Guide.

Future work aims at addressing weaknesses that have been revealed during these applications. This includes technical improvements such as automated report generation to save manual effort, automated statistical analysis to support comparisons between different assessments or tool-supported maintenance to allow simplified adding and removing of assessment criteria.

With respect to the functionality of the tool, assessors would highly appreciate tool support for comparing, in real time, answers in an interview that they are currently conducting to answers and analyses from past assessment interviews. The availability of such an interview tool, which provides a sophisticated guidance mechanism, could reduce follow-up effort required to clarify any issues in further interviews.

In addition to these technical improvements, it would also be worthwhile to invest work into the refinement of questions and answers to further improve assessment results. Such a refinement could be done in various directions. For example, assessment criteria could be extended to questions investigating specific software project characteristics such as compliance to regulatory issues (e.g., medical standards) or criticality of software applications, (e.g., regarding reliability, safety, or security requirements).

Moreover, in the future, the RE Assessment Guide could also be adapted to specific project settings such as RE in systems engineering projects, RE in agile projects, RE in mechatronic projects, RE in near- / offshoring projects, etc.

Because this approach is to some extent a survey approach, it would also be worthwhile to review the research literature with respect to the composition of unbiased survey questions and tune questions systematically.

We also plan to improve the result analysis and report generation capabilities of the RE Assessment Guide. The goal is to make the assessment results better comparable between different companies. This could also mean to refine the rating system itself (e.g. by considering a different weighting of questions and answers).

REFERENCES

- [1] H. F. Hofmann , F. Lehner, "Requirements engineering as a success factor in software projects", IEEE Software, vol. 18, no. 4, pp. 58–66, 2001.
- [2] M. I. Kamata, T. Tamai, "How does requirements quality relate to project success or failure?", in Proc. 15th IEEE International Requirements Engineering Conference (RE'07), pp. 69–78, 2007.
- [3] A. Rainer, T. Hall, N. Baddoo, "Persuading developers to 'buy into' software process improvement: local opinion and empirical evidence", in Proc. 2003 International Symposium on Empirical Software Engineering (ISESE'03), 2003.
- [4] T. Olsson, J. Doerr, T. Koenig, and M. Ehresmann, "A flexible and pragmatic requirements engineering framework for SME", in Proc. 1st International Workshop on Situational Requirements Engineering Processes: Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes (SREP'05), pp. 1–12, 2005.
- [5] W. Bekkers M. Spruit, I. van de Weerd, R. van Vliet, and A. Mahieu, "A situational assessment method for software product management", in Proc. 18th European Conference on Information Systems (ECIS 2010), paper 22, 2010.
- [6] J. P. Kuilboer, N. Ashrafi, "Software process and product improvement: an empirical assessment", Information and Software Technology, vol. 42, no. 1, pp. 27–34, 2000.
- [7] D. J. Reifer, "The CMMI: it's formidable", Journal of Systems and Software, vol. 50, no. 2, pp. 97–98, 2000.
- [8] W. Bekkers and I. van de Weerd, SPM Maturity Matrix, Technical Report UU-CS-2010-013, Utrecht University, 2010.
- [9] S. Beecham, T. Hall, and A. Rainer, "Defining a requirements process improvement model", Software Quality Journal, vol. 13, no. 3, pp. 247–279, 2005.
- [10] E. Geisberger, B. Berenbach, M. Broy, J. Kazmeier, D. Paulish, and A. Rudorfer, Requirements engineering reference model (REM), Technical Report TUM-I0618, Technische Universität München, 2006.
- [11] CMMI Product Team, CMMI® for development, version 1.3, Technical Report CMU/SEI-2010-TR-033, Carnegie Mellon University, 2010.
- [12] ISO/IEC 15504, Information technology – Process assessment, International Organization for Standardization, 2004.
- [13] C. Gresse von Wangenheim, T. Varkoi, and C. F. Salviano, "Standard based software process assessments in small companies", Software Process Improvement and Practice, vol. 11, no. 3, pp. 329–335, 2006.
- [14] J. Dörr, S. Adam, M. Eisenbarth, and M. Ehresmann, "Implementing requirements engineering processes: using cooperative self-assessment and improvement", IEEE Software, vol. 25, no. 3, pp. 71–77, 2008.
- [15] W. Bekkers, I. van de Weerd, S. Brinkkemper and A. Mahieu, "The influence of situational factors in software product management: an empirical study", in: Proc. 2nd International Workshop on Software Product Management (IWSPM'08), pp. 41–48, 2008.
- [16] T. Gorschek and A. M. Davis "Requirements engineering: in search of the dependent variables", Information and Software Technology, vol. 50, no. 1-2, pp. 65–75, 2008.
- [17] T. Gorschek and C. Wohlin, "Requirements abstraction model", Requirements Engineering, vol. 11, no. 1, pp. 79–101, 2006.
- [18] F. Pettersson, M. Ivarsson, T. Gorschek, and P. Öhman, "A practitioner's guide to light weight software process assessment and improvement planning", Journal of Systems and Software, vol. 81, no. 6, pp. 972–995, 2008.
- [19] C. Rupp, Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis (in German), 5th edition, Hanser, Vienna, 2009.
- [20] LimeSurvey Tool, available under <http://www.limesurvey.org> (last access March 10th 2014)
- [21] P. Kruchten, The rational unified process: an introduction, 2nd edition. Addison-Wesley, Boston, 2000.
- [22] <https://assessmentguide.limequery.com/> (for login, please contact authors from Zühlke Management Consultants AG)

Capturing and Sharing Domain Knowledge with Business Rules

Lessons Learned from a Global Software Vendor

Walid Maalej

University of Hamburg

Hamburg, Germany

maalej@informatik.uni-hamburg.de

Smita Ghaisas

Tata Consultancy Services

Pune, India

smita.ghaisas@tcs.com

Abstract—Business rules represent constraints in a domain, which need to be taken into account either during the development or the usage of a system. Motivated by the knowledge reuse potentials when developing systems within the same domain, we studied business rules in a large software company. We interviewed 11 experienced practitioners on how they understand, capture, and use business rules. We also studied the role of business rules in requirements engineering in the host organization. We found that practitioners have a very broad perception for this term, ranging from flows of business processes to directives for calling external system interfaces. We identified 27 types of rules, which are typically captured as a free text in requirements documents and other project documentation. Practitioners stated the need to capture this tacit form of domain knowledge and to trace it to other artifacts as it impacts all activities in a software engineering project. We distill our results in 17 findings and discuss the implications for researchers and practitioners.

Index Terms—Requirements Knowledge, Business Rules, Empirical Studies, Software Documentation, Domain Knowledge

I. INTRODUCTION

Business rules are statements that define or constrain particular aspects of the business [9]. They are often used to specify the behavior of a system under development or how it should be used. Business rules are also used to capture constraints and conditional instructions, such as an allowed input range or an important action that must be performed when an event occurs. From the mid-1990s, the information systems and business-process management communities have discovered the potential of systematically capturing and managing business rules. Several initiatives have focused on formalizing business rules, resulting in either rule definition languages such as RuleML [3], Semantic Web Rule Language (SWRL) [11], and extensions of the Unified Modeling Language (UML) [20], or in tools for defining and maintaining business rules such as SAP Netweaver Business Rules component [17] and Be Informed [19].

In the software and requirements engineering (RE) communities, business rules have been mainly studied with the purpose of developing rule engines and rule-based systems that aim to facilitate the “business driving IT” vision [13]. However, there has been less emphasis on capturing the knowledge contained in business rules and using it to facilitate documentation or communication in software projects. In this

paper, we study how software, in particular requirements engineering practitioners, understand, capture, maintain, and use the knowledge contained in business rules.

Our work was motivated by two observations, which we made in the company that hosted this study. First, we observed in a large insurance project that a minor change in one business rule (a modification in the allowed growth rate of insurance funds as determined by a country law) resulted in a significant ripple effect and maintenance effort, which lasted for 6 months. This rule was described in numerous documents and its implementation was scattered across different parts of the system. Second, we observed that many rules are domain, region, or company specific rather than application specific. Automatically identifying and extracting the rules and tracing them to related project artifacts would increase reuse when developing a new application for the same domain, region, or company [7]. This is the long-term goal of this research.

To achieve automation, we must first study current practices and problems and carefully derive solution strategies. This paper reports on the findings from detailed interviews with eleven practitioners from a global software vendor based in Asia. We asked practitioners from different backgrounds, domains, and roles about their understanding of business rules, the way they capture and maintain rules, and whether they see any problems and potential improvement in current practices. We also studied participants RE practices and problems to put the answers in the overall project contexts.

The remainder of the paper is structured as follows. Section II summarizes previous related work. Section III introduces the study design, including the research questions, research method, and participants. Section IV and Section V summarize our findings: first on business rules, their definition, maintenance, and usage and then on RE practices and problems, which impact business rules. Finally, Section VI discusses the implications of the findings for practitioners and researchers, while Section VII concludes the paper.

II. RELATED WORK

Early works from the information system and business-process management communities focused on the definition, classification, extraction, and management of business rules.

Definition. According to the Business Rules Group definition [9], a business rule either asserts business structure, controls, or influences its behavior. Gottesdiener [8] defines

business rules as declarative, atomic, distinct, business oriented and business owned rules that are expressed in natural language. Ross [16] defines several dozens of atomic rule types and equates them to elements in the periodic table. Ceri and Fraternali [5] claim that business rules model the reaction to events that occur in the real world. Business rules are also considered as a requirement on condition or manipulation of data [18] and as a computational requirement that determines or affects how business is run [15].

Classification: The efforts to classify business rules took various viewpoints into account. Wieden et al. [21] proposed 15 different “semantically-oriented” rule types grouped into structural, behavioral, and managerial categories. Zoet et al. [22] proposed a business rule categorization that is aligned to the business process management lifecycle. Herbst et al. [10] argued that common data-oriented methods are insufficient and inconvenient for a complete modeling of business rules.

Extraction. Researcher previously suggested approaches to extract business rules from structured and unstructured text, motivated by the reuse potential of rules. Ali et al. [1] suggested an approach that takes rule repositories either in relational databases or text format as input and convert it into xml syntax by applying transformation method on SQL queries or a parsing and transformation method using xquery. Mahgoub et al. [12] integrated XML technology with Information Retrieval techniques to automatically select the most discriminative keywords for association rules generation and used Data Mining techniques for association rules discovery. Breaux and Antón [4] proposed a method to mine rule semantics for understanding legislative text.

Management. Spreeuwenberg et al. [19] suggested using controlled natural language and pattern sentences to enable the involvement of domain experts in system modeling and allowing to define and understand business rules. The authors discussed advantages of mapping pattern sentences with the underlying meta-model. They also highlighted challenges in implementing the approach for a larger audience and in dealing with variations of rule sentences. Döhring et al. [6] discussed the use of business rules in combination with an “eventing semantics” to introduce flexibility in workflows. Becker et al. [2] used business rules to model compliance requirements in the financial sector and presented an extension of the semantic process modeling language for managing the rules.

Despite the long-standing recognition for their business impact, business rules remain rather unpopular. Resch [14] concluded the need to know much more about business rules in order to unleash the full potential of rule management systems. The author suggested that research should start with the rationale view and this reveals many questions, which must be answered by sound empirical and experimental research.

The empirical studies that we came across, e.g., by Weiden et al. [21], Zoet et al. [22] or Herbst et al. [10] start with a hypothetical rule classification. None has attempted to probe what the practitioners’ understand when they refer to rules. To our knowledge, there is no published study on how software

and requirements practitioners perceive and maintain business rules. Our work is a step towards understanding this perception.

III. STUDY DESIGN

We summarize the research questions of this study, introduce the method followed, and describe the participants.

A. Research Questions

The main goal of this study was to qualitatively explore how business rules are being used in software and requirements engineering projects and what are recurrent problems and potential improvements. This implies answering the following specific research questions:

- **RQ1:** How are business rules being perceived in practice and which types of business rules exist?
- **RQ2:** How do stakeholders capture and maintain business rules?
- **RQ3:** How are business rules used in software projects?
- **RQ4:** Which impact does RE practices and tools have on the usage and management of business rules?
- **RQ5:** Which problems related to RE affect the management and usage of business rules?

The goal of RQ1 is to eliminate ambiguities about business rules as different stakeholders might understand them differently. Answering RQ2 and RQ3 includes exploring practices, tools, and templates used by practitioners to define, maintain, and use to business rules. The last two research questions focus on exploring the relationship between RE practices and business rules.

B. Research Method

To answer the research questions, we conducted semi-structured interviews with open questions together with a detailed literature survey (see Section II). We structured the interview questions along three sections:

1. About You: In this section, which lasted for about 10 minutes, we asked the participants to introduce themselves, their backgrounds, and experience. Moreover, we asked about the domain they work in, typical projects, customers, and team characteristics.
2. Requirements Engineering: In this section, we asked the participants about common requirements engineering practices, which tools are used, and whether there are problems related to these practices. This section lasted for about 20 minutes.
3. Business Rules: The last and longest section (60-70 minutes) was dedicated to business rules. We focused on how participants define a business rule and which variants they know, asking for example and experiences. We also asked about maintaining and using the rules and whether there are problems related to current practices and tools. In this section, we reflected on participants answers from Section 2 to put them into the RE context.

TABLE I. OVERVIEW ABOUT INTERVIEW PARTICIPANTS

| P# | Roles | Exper. (yrs) | Projects characteristics | Team | Domain |
|-----|--|--------------|---|--|---------------------------------------|
| P1 | Domain consultant, business process manager, requirements reviewer | 13 | Large programs (6 years) on property and casualty insurance, personal insurance for a large company | 1500 people at peak time. 80 in RE | Insurance |
| P2 | Business process manager, domain consultant, program manager | 20 | Large nationwide insurance deployment for a European government, over 22 months | 350 people in Asia and Europe | Insurance |
| P3 | Developer, project manager, requirement analyst | 4 | Online services using specified technology stacks | 30 people, 2-3 for requirements documentation and analysis | Software Industry, Investment Banking |
| P4 | Developer, maintenance lead, project manager | 20 | Development of an electronic shares trading application, Business intelligence, reverse engineering, maintenance for an insurance company | Ranging from 100-300 in varied locations | Telecom, Insurance, Power |
| P5 | Developer, software architect, project manager | 12 | Documents and claims management for public institutions, Customer Relationship management for telecom companies | Ranging from 50-100 people | Finance, Telecom |
| P6 | Software architect, program manager, delivery manager | 10 | Infrastructure software for large telecom companies across the world, a large automotive company | 40-120 people | Automotive and Telecom |
| P7 | Program manager, product manager | 16 | Development and customization of a market infrastructure with 45 components, 5 offerings for numerous large banks | 30-50 people | Financial Services |
| P8 | Project manager, program manager | 21 | Product customization and delivery for seven banks | Ranges from 7/8 to 200/300 depends on extent of customization to be done | Financial Services |
| P9 | Product manager, presales lead | 17 | Customization of a financial services infrastructure | 1 pm to 100 person months | Banking and Financial Services |
| P10 | Tester, lead underwriter, program manager | 14 | Testing product suite for large banks in US | 30-35 people | Banking and Financial Services |
| P11 | Delivery center head | 18 | Large program management | 30-90 people | Financial Services |

The full list of the interview questions is available online at http://www.teamweaver.org/wiki/index.php?title=Business_rules_interviews Each interview lasted for about 90 minutes and was carried out either via phone or face-to-face. While one main interviewer (one of the authors) moderated the interview sessions, one or two other interviewers were present to take notes and ask for clarifications. This reduced the interviewer's bias. The main interviewer summarized the minutes along the questions within 48 hours. Then the other interviewer(s) and the participants were able to refine and extend the minutes. For the summary of the results, a statement that was observed twice was included and extended iteratively with quotes from other minutes.

To summarize the results, we first aggregated the statements along the research questions. Then, the authors independently tagged the answers of the participants and grouped quotes that seemed similar. In the subsequent iteration, we discussed the groups of the quotes and merged them. Finally, we revisited the interview minutes, identified similar statements, and merged them into the findings while taking care of preserving their meanings.

C. Participants

Between June and September 2012, we interviewed 11 participants. The selection of the participants was based on two criteria. First, we aimed at getting participants from different – at least two – business units (i.e. industrial sectors) in the host

organization. Second, we aimed at capturing different project roles and people with different background, and varied experience to increase the validity of our results. Table I summarizes the participants' details. Several participants, e.g., P2, P6, P7, and P9, had also worked in other organizations and reported on their experience in general.

IV. FINDINGS ON BUSINESS RULES

We summarize the results of RQ1, RQ2, and RQ3.

A. Definition of Business Rules and their Types

Finding 1: Stakeholders have different perceptions of business rules depending on their roles and experience.

While all participants agreed that business rules usually represent constraints and restrictions, there was a disagreement about the nature of restrictions and the characteristics of the business rules. For instance, P2 stated that business rules are constraints for a system that are *non-negotiable* and are driven by corporate policy or regulations. For example, investments in a single fund cannot exceed N% of total investment. P11 said that "repetitiveness" is a way to recognize rules. He cited the example of auctions at central banks. Any auction has a start-time, end-time, cut-off time, and weekly frequency. These parameters "repeat for each central bank with different values."

We noticed that a group of participants (P3, P4, and P8) focused more on the *system and product perspective* when describing business rules. They considered business rules as

TABLE II. TYPES OF BUSINESS RULES MENTIONED IN THE INTERVIEWS

| ID | Type | P1 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | Stated # |
|----|------------------------------------|----|----|----|----|----|----|----|----|----|----|-----|-----|----------|
| 1 | Validations rules & value ranges | | | x | x | x | x | | | x | x | | x | 7 |
| 2 | System/ application specific rules | x | x | x | | | x | | | | | x | | 4 |
| 3 | Calculation rules | | | x | x | x | | | | | | | x | 4 |
| 4 | Access control rules | | | x | x | | | | | | | x | x | 4 |
| 5 | External system interfaces | | | x | | x | x | | | | | | | 3 |
| 6 | Laws & regulations | | | | x | | x | | | | | x | | 3 |
| 7 | Sequencing/ control flow | | | | x | | | | | x | | x | | 3 |
| 8 | Geography specific rules | | | | | | | x | | | x | x | | 3 |
| 9 | Business process rules | x | x | x | | | | | | | | | | 3 |
| 10 | Procedural/ Operational rules | x | x | | | | | | | | | x | | 3 |
| 11 | User interface rules | | | x | x | | | | | | | | | 2 |
| 12 | Company policies | | | | x | | | | | | | x | | 2 |
| 13 | Time restrictions | | | | | x | | | x | | | | | 2 |
| 14 | Dynamic rules | | | | | | | | x | | x | | | 2 |
| 15 | Definitions | x | x | | | | | | | | | | | 2 |
| 16 | Data rules | | | x | | | | | | | | | | 1 |
| 17 | Non-functional rules | | | x | | | | | | | | | | 1 |
| 18 | Boundary scenarios | | | x | | | | | | | | | | 1 |
| 19 | Variation rules | | | x | | | | | | | | | | 1 |
| 20 | Ranking of items | | | | x | | | | | | | | | 1 |
| 21 | Mandatory constraints | | | | x | | | | | | | | | 1 |
| 22 | Industry standards | | | | | x | | | | | | | | 1 |
| 23 | Exceptions to known exceptions | | | | | x | | | | | | | | 1 |
| 24 | Usability rules | | | | | | x | | | | | | | 1 |
| 25 | Configuration rules | | | | | | | | x | | | | | 1 |
| 26 | Domain specific rules | | | | | | | | | | x | | | 1 |
| 27 | Dependency rules | | | | | | | | | | x | | | 1 |

conditions or constraints that are mandatory and affect certain product features. P4 compared a business rule to a software component, which has an input, an output, and some processing. The processing part typically must follow some rules. The input has certain conditions and restrictions. These correspond to the business rules. He gave the example of standards in coloring financial charts, depending on ranges of values, system interfaces, or how a system should behave. P8 focused more on the product perspective. He stated, “A business rule is anything that characterizes a product and *how* it works, such as screen behavior or batch processes.”

Other participants had a more general understanding of rules: either from a *requirements* (P5, P9) or a *business process* perspective (P1, P7). P5 explained, “If the customer should, e.g., be able to modify her account, the corresponding business rule is to allow customer to modify some fields.” P9 even claimed that every *requirement* is a high level rule “e.g., this process works this way in this geography.” P1, P2, and P7 mainly reduced business rules to restrictions on *process* models. P7 stated, “A rule is an encapsulation of some aspect

of how a business objective is to be achieved, e.g., trade acceptance should happen between 9am to 5 pm.”

Finding 2: Types of information that practitioners consider as business rules are listed in Table II.

When asking about the types, participants mentioned in total 27 types of rules. Table II lists all types of business rules that were mentioned in the interviews and by how many participants. These types are not mutually exclusive and might overlap. The most frequently stated type was validation rules. An example stated by P3 was “A particular functionality should not be available on Sunday.” The second most frequent type was system/application specific rules (e.g. restricting the deployment of the application and how it should be used), followed by calculation rules (i.e. how to calculate a particular value), access control rules, rules to use external system interfaces, and laws & regulations.

We were surprised that definitions, non-functional rules, and flows were stated as business rule types, as we expected them to be separate requirements concepts. P1 explained that definition rules are like concept specification, e.g., detailing

what is a vehicle for the insurance domain and which different types exist. Like operational rules and definitions rules are never implemented but are important for the comprehension and implementation of other rules" (P2). Non-functional rules include availability restrictions, performance rules, and the number of concurrent users at time. As example of a usability rule, P5 stated the US law that public systems must support people with disabilities. Sequencing/ control flow restricts the functionality of the system, its behavior or the process, e.g. "Unless users fill in the primary details, don't show them the screen to enter other secondary details."

B. Capturing and Maintaining Business Rules

Finding 3: Business rules are embedded as free text anywhere in the project and domain documentation.

All practitioners reported that rules are not necessarily documented in separate documents or separate sections in a document. They can be related to and embedded in a business process, a process step, functionality, a feature, a screen, a mockup, or a system component. For instance, P5 stated that there is "no separate section for business rules", and that these are "captured as part of functionality." P3 stated, "We can find rules as descriptions of features or a set of validation constraints." P1 and P2 stated that rules can be found in requirements specifications, source documents, process description, marketing brochures, regulations (such as telecommunication standards), or laws.

Participants mentioned a few exceptions. P6 said, "Rules are mainly documented in free text. Only in exceptional, safety critical cases they are modeled or formalized." P4 said, "At most in embedded systems project, they try to formally document rules to conduct formal verifications. This is done only if people's life is concerned." P2 stated, "A separate rule sheet is maintained only if the customer insists on this. Otherwise, process steps and rules are not segregated."

One participant (P1) mentioned a tool, which was introduced for maintaining rules. He said, "At the beginning we did not capture rules systematically. Later a new task force was introduced to systematize maintenance of rules. We decided to document and maintain the rules in a database." He argued that this helped to deal with the complexity of the domain. Moreover, the team originally envisioned an automatic maintenance of the system by maintaining the rules. However, the participant admitted that this never succeeded due to the new complexity introduced by formalizing the rules.

Finding 4: Stakeholders rely on assumptions instead of externalized business rules.

Six participants stated that, in particular in the early project phases, many business rules remain as assumptions and are neither captured nor discussed. P8 said, "We relied on a lot of assumptions. No business rules were in place, Actions and validations were not documented. A customer wanted to know the validations, but they were neither documented nor implemented. So it was a mess."

P5 explained, "Gaps between what is implemented and what the customer actually wants are rooted in missing rules.

These are not captured until we get change requests and the actual development starts." P4 stated, "Constraints on how systems interact with each other are very critical in the telecom domain and often unknown or assumed. Even the customer cannot say much about this." P9 agreed that this applies to the banking domain too and explained, "To find out these rules we need to understand the system, try out its interfaces, what are the conditions, which authorization, and access control are there, when the data is available, which ranges are allowed etc. Many times these problems arise in the client acceptance test."

Finding 5: The use of rule languages and rule engines is rather exceptional in practice.

All participants except P1 agreed that no rule language or rule engine was found to be relevant, useful, or essential in practice. The reasons quoted were as follows: (1) the learning effort associated with formal languages, rule engines, and tools; (2) the lack of the "right" people to specify and formalize the rules; (3) the unwillingness of customer to try new things since "they view this as a risk"; (4) the cost of licenses; (5) the resistance to change; (6) the lack of tools (or time to identify the right tools); and (7) the maintenance effort of formal rules which might over-exceed the maintenance effort of source code. P1 stated that a tool (RuleXpress) was piloted during a project to build a repository of rules and maintain them. In his opinion, this was helpful as the rules were more precise.

C. Usage of Business Rules

Finding 6: Business rules are captured and used not only in requirements engineering but also in other project activities.

There was a general agreement among participants that rules get "discovered" and used in all phases of a project. While some participants (e.g. P2) claimed that everyone used rules without realizing, others (e.g. P6) were more specific in stating that rules were discovered during RE and used in implementation and testing phases. P7 claimed that "many rules were discovered during the implementation, and testing and were used in these phases as well." P9 explained that her team used rules for creating and updating product documentation. She also claimed that product configuration is "rule-driven" and enhancements requests were often about adding new rules. P8 concurred that product management teams look at rules to detect commonalities and "highlight them" for the next implementation.

Finding 7: If captured correctly, business rules can reduce the complexity of domains and implementations.

Seven participants (P1, P6, and P7-P11) expressed that the complexity of a domain can be simplified with the definition of business rules. P1 informed that the underwriter team in his project examined the code and found 150 rules implemented. The team then analyzed these rules and identified much overlap and unneeded complexity. They were able to reduce the 150 to only 12 rules. This was a "big saving, due to better business efficiency and easier maintenance of the code." P6 gave an example from the Telecom domain. When an SMS is sent, the system has to decide what message to be sent to whom, when,

and whether it should wait. Decisions about simultaneous notifications to multiple entities need to be made. All these aspects are captured in rules.” Handling failure of an event (such as an SMS) is also determined by a set of rules. P11 cited an example from the banking domain. For fixed deposits there are global and local rules regarding periodicity, interest for different products, customer-specificity, and 30X360 rate factors in fixed bonds. All of these get encapsulated “nicely in rules.” P7 gave an example of the market infrastructure for the banking domain. User behavior, conformance to current trend, user preferences, and country specifics are increasingly getting encapsulated in rules, similarly to how to dynamically handle peak loads of trade transactions.

Finding 8: Externalized business rules facilitate sharing knowledge with new team members.

In the host organization, people frequently move between units and projects. Five participants (P1, P2, P3, P4, P7) mentioned that business rules offer a concise mechanism to communicate existing requirements knowledge to new entrants in a team.

P1 expressed that in every annual review new people joining the project needed a 3-4 weeks of intensive training. After rule documentation, the time was reduced significantly to just 1 week. P3 claimed, “People who join projects later have big problems to read and understand 400 plus pages documents.” P4 found that new people joining must understand how the whole system is supposed to work, but have very little time available for this task due to delivery deadlines. If business rules were documented, this process would be easier.

Finding 9: Externalized business rules facilitate the reuse of domain knowledge across the projects.

The organization that hosted this study develops thousands of systems over the years in the same domain for different customers across the globe. Naturally, there is a lot of emphasis on domain knowledge management for reuse to achieve productivity and efficiency benefits [8]. Reuse of existing domain knowledge, which was acquired and validated in previous projects allows for a needed quick-start to projects and ensures good quality requirements. This is especially useful, given that the time allotted to requirements is short.

This was mentioned explicitly by two participants (P1, P6) and was implied in the interviews of four (P7, P8, P9, P11). P1 stated that the insurance domain is tightly regulated in the US with small variations across the states. Therefore there is a large potential for reuse across the projects developed for different states. P6 expressed that the reuse of rules depends on similarity of purpose. If a rule is generic enough to be applicable across, it may be used across the domain or company. P7 stated that understanding a complex domain was “inadequate in the absence of articulation of rules.” P11 mentioned the bonds auction process had many similarities across the world and the common rules were amenable to reuse. P9 mentioned the regulatory challenges of the financial services domain when it comes to variations regulations in different countries while executing the same operations.

V. FINDINGS ON REQUIREMENTS ENGINEERING

This section summarizes our findings on RE practices, tools, and problems related to business rule usage (which correspond to RQ3 and RQ4).

A. Impact of RE Practices and Tools on Business Rules

Finding 10: Business rules are documented differently in service-oriented and product-oriented RE. Product-oriented teams spend more effort to maintain business rules as they are more interested in reuse.

The first half of participants (P1-P6) reported their RE to be service-oriented while the other half (P7-P11) practice a product-oriented RE. In the service-oriented projects, RE was strongly dependent on customers. P6 claimed, “Requirements are not only given but also *driven* by customers” referring to the process which should be followed or the documentation that should be written, including business rules. P5 said, “Customers decide about requirements. They want to understand what they need.” P4 stated, “We focus on business processes of the customer. In the requirements documents we put which steps are parts of the process, what is the input and output of each step, and sometimes the processing rules.” P2 claimed, “We cannot change a customer’s perspective. It starts with defining the approach, templates, plan, and deployment of most suitable subject matter experts vs. available experts, clients’ familiarity of certain tools and methodologies, their existing contract with certain 3rd party vendors for tools with whom there is an ongoing license or contract. All these impact the requirements and the tools.”

In the product-oriented RE (telecom and banking domains) participants’ teams (i.e. the software vendor) initiated the RE process. When these teams are capturing requirements, their objective is “also to sell as many product features as possible” (P8) and to minimize customization effort of existing product platforms. Product-oriented teams seem to update requirement documents more systematically and try to achieve completeness of requirements and identify generic features that must go into a product line. P7 stated, “We have 45 components in the platform. Each component has its own product specification. Each specification describes the functionality of that component in Use Case format including restrictions and rules. We create Business Requirement documentation specifically to each customer. The extended product specification document is created based on a gap analysis. We use our working software as a source for identifying more product requirements and evolve the product and its documentation.” For the service-oriented projects, we did not observe such motivation to update requirements documents. The first priority is to deliver the expected software features on time with the expected quality (P3).

Finding 11: Customers expect software vendors to have domain knowledge and to “develop” business rules.

We also observed that the interviewed teams often play the role of domain experts, even when they did not have previous experience with similar projects. For example, P1 stated, “The customer mainly intervenes in the conflict resolution. We do

the “creativity” on what should be developed and the customer decide in case of conflicts.” P11 claimed, “Customers are unable to give requirements. They give bits and pieces and we have a full-fledged analysis team to do this.”

The participants’ teams had to acquire or “improvise” domain knowledge, e.g., by hiring subject matter experts or by studying public knowledge such as standards and laws. We observed this particularly in the insurance and banking domain, where the regulations change frequently and depend on the location of the customers. Even in technology and IT domains, where software development belongs to their core business, customers tend to “outsource” RE to the software vendor, as claimed by P3 and P5.

Finding 12: There is a trend towards agile projects, which lead to an increased documentation and evolution of rules in natural text documents.

When asking about processes, participants reported to follow both waterfall and agile methods, with a growing trend toward agile (P1, P3, and P6). Customers are increasingly asking for agile delivery because of (1) the “pressure” to follow industry trends, (2) iterative prototyping that allows more frequently for a “sneak peek” into the progress of the project, and (3) the flexibility for prioritizing differently if necessary. Customers may or may not be knowledgeable on agile, but they are keen that vendors adopt agile practices anyway. P3 stated, “We have followed scrum with sprints of 9-13 days. But within a sprint, we followed a waterfall model. We had 2-3 days to read the document, 4 days to implement the features, and 2-3 days for the testing.” P6 stated, “In agile, user story embeds both rules and functionality in a single statement. In waterfall or V model, the Use Case templates may contain separate placeholders such as pre-conditions and post-conditions for capturing rules.”

Finding 13: In industry, office tools are widely used for RE also to capture and maintain rules as part of requirements.

When asking about the tools, nine participants (all except P1 and P6) reported that no specific tools were used for requirements engineering. In most of the projects, participants’ teams used text editors and spreadsheet to capture and manage requirements and rules. When asked about the rationale, P2 and P11 stated, “This is the fastest and easiest way. We use documents and complex spreadsheets since customers are comfortable with them.” P2 and P9 claimed similarly “otherwise, we need tool training. This is time consuming, and licenses are expensive.” P8 claimed, “We do not use particular tools for capturing requirements and rules, if you know something good, please tell us.”

Two participants (P1 and P6) stated that they have used DOORS in a few projects mainly because it supports version control. Other participants used change-tracking mode of Microsoft Word or an extra column in a spreadsheet for version control. P2 claimed, “Spreadsheet did not work well when adjustments were needed.” He referred to the change requests that impact various requirement elements. For example, a change in workflow would require changes all the way from screen navigation to the processes and rules that govern the

processes and it was not easy to track this using spreadsheet. P1 and P6 said that they have used special tools (including ReqPro and Caliber) only if the customer explicitly asked for it. Otherwise, the team reused templates publicly available on the Internet, and sometimes developed own templates in house.

B. RE Problems Affecting Business Rules

Finding 14: Requirement practitioners allocate low effort for understanding requirements, in particular for understanding special flows and business rules.

Six participants (P1, P3, P4, P5, P8, and P10) stated that the time and effort allotted for RE is usually short. This might have a serious impact on the understanding and on the quality of requirements, as well as the planning and the effort estimation.

P3 stated, “Understanding of the requirements is the main thing. This is more important than the documentation. That’s why we need all the discussions.” Participants claimed that due to time constraints there isn’t enough documentation. Sometimes meeting notes are used as a formal requirements document. People who joined the project from the beginning know how it works. People who joined later have big problems to read and understand “400 pages documents” (P3). P5 claimed, “There are very few available resources to do proper requirements engineering. In one large project, the real development started during testing, since all major requirement changes came by then. We needed nine months implementation after starting the system testing. In a project of nine months, at best 1 month is given for requirements engineering. Teams have rush, both in term of coverage and depth of the requirements.”

P8 stated, “At the sign off of the Business Requirements Specifications one customer from North America asked us to share the specifications. When we sent the specifications, we got numerous comments. We had already started programming in parallel. But because of these comments, we had to rework the code and a 6 months project went to 18 months. This happened because the time spent on understanding requirements and business rules was too short. We relied on a lot of *assumptions*, and we did not capture the business rules, the actions, and validations. The customer wanted to know the validation rules but these were undocumented.”

When we asked why less time was allotted to RE participants argued that in most of the projects the rollout deadlines are fixed, either because the customer promises a roll out date to the users, a regulatory deadline must be met, or for competition reasons. In contrast, deciding about the vendor is a long process, which results in a late contract award. The time available for software development is hence shortened. The unrealistic deadlines lead to shortening the entire project. The time allocated for RE is often shortened disproportionately. Within RE, typically restrictions, constraints, special cases, and exceptions are either ignored or not captured and understood.

Finding 15: The lack of domain knowledge in development teams hinders the capturing and usage of business rules as well as reuse across projects.

Five participants (P1, P2, P4, P6, and P9) explicitly mentioned the lack of domain knowledge as a recurring issue in their

software engineering projects. P4 explained, “In our projects, people who do RE do not necessarily know much about the domain. In one extreme case, even the customer was new to the domain. They approached to us because we had previously implemented a similar system. But unfortunately our people who worked on that project had left already.” P2 stated, “The lack of domain knowledge and domain training is a problem while defining and implementing the rules.”

We observed that the number of team members who know the domain well is rather small (P1: 80 from a team of 1500, P3: 2 in a team of 30 people, overall about 5%). Teams seem to have an issue with the optimal utilization of subject matter experts. P1 explained the importance of domain knowledge with an example from the insurance domain in the US, which is tightly regulated. On the one hand, domain expertise is needed to deliver software, e.g., taking into consideration the variations across states. On the other hand, there is a lot of potential for reuse across the projects. We also observed similar claims in other domains. For example, P9 stated, “The financial services market infrastructure domain is very challenging. There are new regulations and business rules all the time. We are not privy to the discussions that happen during these changes.”

Finding 16: Identifying the right level of detail in requirements documentation is difficult. Either too much information is captured and the document becomes too large and unreadable or only main flows without business rules and exceptions are captured and the document becomes useless.

Participants complained that requirements documents do not have the right level of details: either too much information or too little. In particular four participants (P3, P4, P5, P7) claimed that the exceptional flows are not captured in sufficient detail during RE. For example, P5 explained “there is often a narrow view of requirements to just capture *the happy flow*. This often matches the as-is, but not the to-be. We had once, for example, big data migration issues since we had the data structure in place but did not know the workflows and the restrictions on the data.” P4 and P9 gave an example of superficial requirements “calculate a rate based on interest rate.” When starting the implementation based on this requirements developers encounter many questions and ask the people about the details and the rules behind. Teams often had delays in the implementation schedule, because they did not have the required level of details at the beginning. P3 explained “The customers or the managers show you a screen and ask for estimation, which was correct since the details like the specific scenarios, exceptional flows, and the constraints in particular contexts were unknown.”

Surprisingly, participants also complained about too detailed specifications. For instance, P3 complained about the difficulty of finding the right information under time pressure within a 400 pages requirements specification document. P7 stated, “In western countries customers expect high level of requirements details with a very fine-grained change management. In developing countries, organizations are rather small with 20-25 people. For them, over-documentation is overwhelming.” When asking for details, we found that the real

problem lays in the information identification and the efficient answering of stakeholders’ questions due to the static nature of large documents.

Finding 17: Business rules and exceptions represent a substantial amount of requirement knowledge, which often remains tacit in the mind of people.

Five participants (P3, P4, P6, P7, P9) complained that relevant requirements knowledge such as *exceptions*, *rules*, and *justifications* remain tacit in the mind of people. For instance, P4 explained, “Many requirements and rules are implicit. Either the customers do not know it (e.g. interfaces of a legacy system) or the customer thinks you should know it anyway (e.g. the response time or the authorization). Participants explained that knowledge “is there” for years leading to many implicit requirements. P3 stated, “There are implicit rules that are not mentioned anywhere. For instance, the validations for similar fields in web forms are described only once.”

Three participants (P6, P7, and P9) associated this issue with the personal backgrounds and skills of the involved stakeholders. P6 stated, “Technical experts are not involved in requirements exercises. This leaves huge gaps in understanding and interpreting requirements.” P7 proposed, “We have to de-personalize the problems. It depends who does requirements.” As a result of tacit knowledge residing with people, the whole RE process becomes heavily dependent on people. Participants claimed that they would like to make this less subjective and less people-dependent, by making the tacit knowledge explicitly available. They argue that the quality of requirements that are backed with externalized knowledge will be higher than when requirements are based on tacit knowledge.

VI. DISCUSSION

A. Implications

1) Tools for Managing Requirements and Business Rules

One consistent finding of our study is that requirements tools are rarely used in practice – at least in the host organization. The participants were aware of special RE tools but they used them neither to capture business rules nor other types of requirements knowledge. Given the size of this organization (over 100.000 employees) and the broad spectrum of domains and customers involved, we think that this finding should be taken seriously into consideration by requirements engineering researchers and tool vendors.

Over two decades of tool research and development word processors and spreadsheet tools are predominating RE in the studied projects. As consequence, information is difficult to retrieve and detailed questions cannot be easily answered (Finding 16). A frequently mentioned example in the interviews was of the static requirements document of 400 pages, that no one wants to read or maintain. We wonder: When do we need RE tools at all? When do we need formal means for capturing business rules and other types of requirements knowledge? Why are other software engineering tools, such as bug trackers or IDEs, more established amongst practitioners? To answer these questions, more systematic, broad, and reliable studies of RE practices are needed.

There are pragmatic and fundamental reasons behind this situation. Pragmatic reasons for using spreadsheets and office tools for capturing rules and domain knowledge in general include license costs, training effort, compatibility, and usability issues. Fundamental reasons include the way state-of-the-art requirements tools support capturing, updating, and accessing requirement information especially in large projects. Participants claimed that relevant information is available somewhere. However, it is difficult to get the right information in the right context. Usually stakeholders must read hundreds of pages manually and reason about interdependencies: sometimes having in mind that the requirement document might be not up-to-date. Researchers should study means to automatically mine types of business rules in text and develop question-answering systems to retrieve the required information with the required details depending of the current task and context. This by itself would increase the usefulness of domain knowledge and business rules and provides an incentive to capture and maintain them.

2) Reuse of Domain Knowledge Through Business Rules

One main lesson learnt from this study is that nowadays software organizations do not only rely on the asset of the software, technology, and design knowledge, but also more and more on the domain knowledge in general and business rules in particular. We learnt that users and customers expect engineering teams to know the domain very well. And if not, it was not easy for the teams to extract the characteristics of the domain, the restrictions, the constraints, and the exceptions. Customers either assume that domain knowledge is common or they have no detailed knowledge about the domain.

We learned that business rules represent a crucial part of domain knowledge, but unfortunately frequently reside implicit in the mind of subject matter experts. If captured or extracted, business rules would bring large potential for reuse. Business rules can be found as a natural text in any project document from a marketing brochure, to a requirements document, an email, or a source code comment. Extracting and tracing them to other artifacts can assist requirements stakeholders while working on different tasks: from the negotiation of requirements to the planning and testing.

To this end, the first step is to get a common understanding of the types and knowledge included in business rules. In this study, we found that there is no precise common definition of the term business rule amongst practitioners. However, there exist an implicit understanding. All practitioners knew the term and were able to give examples and report on their experience. We identified 27 types of business rules. Some of them were mentioned frequently by the participants. Some types strongly overlap with other types of software engineering knowledge, which can be found, e.g., in API documentation or manuals. This increases the reuse potential of business rules.

Our results let us believe that researchers and tool vendors should focus more on the knowledge and documentation potentials of business rules in addition to the automation and code generation potentials from rule based engines. For instance, business rules can be included as annotations to accelerate understanding of complex documents. The system

design can benefit from highlighting rules in a document. Search engines can crawl the various artifacts and leverage business rules to stakeholders through search interfaces. One participant mentioned, "It would be perfect if I could just select a component or a feature and my tool show me the most important business rules which I should take care of." To realize this vision, the software and requirements engineering research community need to learn from the information retrieval or knowledge management communities.

B. Limitations and Threats to Validity

There are several limitations to the internal and external validity of our results. We are aware that in 90 minutes we can only discuss a fraction of participants' activities and experience. We might have missed certain types of business rules, usage scenarios, or problems. However, we think that extending the interview time would not fundamentally change the findings since concentration typically decays over time. Moreover, participants were able to give free comments at the end the interviews or contact us afterwards via email or phone.

Another potential threat to the internal validity is that interviewers might have had assumptions and expectations and might have reported only clues affirming these expectations, while ignoring different unexpected statements. To mitigate this threat, all interviews were conducted by at least two interviewers with a rotating minute taker. The minutes were also circulated to the interviewees to validate the results and add clarifications.

Similarly, participants might have behaved differently because they were interviewed, stating what the interviewers "are willing to hear." This threat cannot be eliminated completely but we addressed it by assuring participants complete anonymity and confidentiality. We also stressed that there was no "right and wrong answers" as we only aimed at documenting the state of practice through their experience and subjective opinions. Moreover, we only report on findings that were observed at least in two different interview sessions.

We also asked participants to send example documents that backup important examples and statements. We received these documents from 8 participants. Nevertheless, observing the participants during their daily work will lead to complimentary, or possibly different results and additional evidence. Finally, our sample size with 11 participants is rather small. Because we did not study a random sample, that is representative of the entire target population of stakeholders, it is difficult to generalize from these findings. But, our study was designed to be exploratory, qualitative rather than representative, quantitative. It has a strong degree of realism as it was conducted with real industrial experts.

Within the host organization, we were unable to draw representative samples from all employees due to the coordination and interviewing effort. However, we sampled exclusively experienced participants, all experts within their teams. The distribution of participants includes different roles, application domains, projects, and technologies – representing a wide range of potential participants. We purposefully recruited at least two people from each participating domain (insurance, banking, telecom, and industry). This gives us some

confidence that the results have a medium degree of generalizability – at least within the target organization and similar outsourcing companies and IT service providers. The quantifications reported in this paper (e.g. the number of observations) should be interpreted carefully. The number of observations and statements might seem low at the first glance. We think however that all reported statements are important since participants brought them by their own as opposed to surveys were options are given to participants. Nevertheless, checking our findings and quantifying the results would require conducting a broad online survey and content analysis studies with representative samples.

VII. CONCLUSION

In this paper, we studied how business rules are being used in requirements engineering practice. We identified 17 findings describing the types of business rules, the way they are captured and maintained, as well as problems in managing and using them. Experienced, interviewed employees of a large software vendor and IT service provider agreed that capturing and managing business rules is important and would impact not only requirements engineering activities but also other activities such as testing, documentation, and planning. However, neither the current RE tools, nor processes give room to easily retrieve business rules, trace them to other project artifacts, or reuse them in future projects within the same domain or for the same customer. To support practitioners dealing with these challenges the research community should move from exploratory to explanatory and evaluative research, focusing on systematically identifying, mining, and (re)using business rules as a main part of domain knowledge.

ACKNOWLEDGMENT

The TCS Research Sabbatical program enabled this collaboration. We are very grateful for Preethu Rose for the support with conducting and summarizing the interviews as well as all participants for their effort and feedback.

REFERENCES

- [1] S. Ali, B. Soh, and J. Lai. Rule extraction methodology by using XML for business rules documentation. In *3rd IEEE International Conference on Industrial Informatics*, 2005.
- [2] J. Becker, C. Ahrendt, A. Coners, B. Weiß, and A. Winkelmann. Modeling and Analysis of Business Process Compliance. In *Governance and Sustainability in information systems managing the transfer and diffusion of IT*. IFIP Advances in Information and Communication Technology, Vol. 366. Springer 2011.
- [3] H. Boley, B. Grosof, and S. Tabet. RuleML Tutorial. The RuleML Initiative, 2005. Last visited Jun. 2014
<http://www.ruleml.org/papers/tutorial-ruleml.html>
- [4] T. Breaux, and A.I. Antón. Mining Rule Semantics to Understand Legislative Compliance. In *WPES '05 Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pp. 51-54, ACM, 2005.
- [5] S. Ceri and P. Fraternale. Designing Database Applications with Objects and Rules, The IDEA Methodology. Addison and Wesley, 1997.
- [6] M. Döhring, B. Zimmermann, and E. Godehardt. Extended Workflow Flexibility Using Rule-Based Adaptation Patterns With Eventing Semantics. In *40. GI Jahrestagung Informatik 2010, Service Science*, LNI, GI, pp.216-226, 2010.
- [7] S. Ghaisas and N. Ajmeri. Knowledge-Assisted Ontology-Based Requirements Evolution. In W. Maalej and A. K. Thurimella (ed.) *Managing Requirements Knowledge* pp. 143-167, Springer 2013.
- [8] E. Gottesdiener. Business RULES - Show Power, Promise. EBG Consulting, Inc. Application Development Trends 4 pp.3.1997.
- [9] D. Hay and K. Healy. Defining Business Rules - What are they Really? The Business Rule Group, 2000.
- [10] H. Herbst, G. Knolmayer, T. Myrach, and M. Schlesinger. The Specification of BusinessRules: A Comparison of selected Methodologies. In A.A. Verijn-Stuart, T.-W. Olle (ed.) *Methods and Associated Tools for the Information System Life Cycle*. pp. 29-46, Elsevier 1994.
- [11] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 2004. <http://www.w3.org/Submission/SWRL/>
- [12] H. Mahgoub, D. Rösner, N. Ismail, and F. Torkey. A Text Mining Technique Using Association Rules Extraction. In *International Journal of Computational Intelligence*, Vol. 4 Issue 1, p21. 2008.
- [13] C. Potts. Let the Business Drive IT Strategy. In *cio.com* (online) Sep. 2007. Last visited Jun. 2014.
http://www.cio.com/article/135202/Let_the_Business_Drive_IT_Strategy
- [14] O. Resch. Six Views on the Business Rule Management System. In *E-Journal of Practical Business Research*. 11/2010.
- [15] D. Rosca, S. Greenspan, M. Feblowitz, and C. Wild. A Decision Making Methodology in Support of the Business Rules Life Cycle. In *Proceedings of 3rd IEEE International Symposium on Requirements Engineering (RE'97)*, IEEE Computer Society Press, pp. 236-246, 1997.
- [16] R. Ross. The Business Rule Book: Classifying, Defining and Modeling Rules. Database ResearchGroup, Boston, MA, 2nd edition, 1997
- [17] SAP Solution Brief. Centrally Compose, Execute And Maintain Business Rules. Last visited on Jun. 2014.
<http://scn.sap.com/docs/DOC-4570>
- [18] P. G. Sellfridge, R. C. Waters, and E. Chikofssky. Challenges To The Field Of Reverse Engineering. In *Proceedings of Working Conference on Reverse Engineering, WCRE '93*, IEEE Computer Society Press, pp.144-150.1993.
- [19] S. Spreeuwenberg, J. Von Grondelle, R. Heller, and G. Grijzen. Using CNL Techniques And Pattern Sentences To Involve Domain Experts In Modeling. *Lecture Notes in Computer Science*, V. 7175, pp.175-193, 2012.
- [20] P. Vincent. OCEB White paper on Business Rules, Decisions and PRR. OMG 2008. Last visited Jun. 2014.
<http://www.omg.org/oceb/OCEB-WhitePaperforPRR1.1.pdf>
- [21] M. Weiden, L. Hermans, G. Schreiber, and S. Van der Zee. Classification and Representation of Business Rules, Universiteit van Amsterdam, 2002.
- [22] M. Zoet, J. Versendaal, P. Ravesteyn, and R. Welke. Alignment of Business Process Management and Business Rules. In *European Conference on Information Systems (ECIS)*, 2011.

Building a National E-Service using *Sentire*

Experience Report on the Use of Sentire: A Volere-Based Requirements Framework Driven by Calibrated Personas and Simulated User Feedback

Chris Porter

Dept. of Computer Science
University College London
London, United Kingdom
c.porter@cs.ucl.ac.uk

Emmanuel Letier

Dept. of Computer Science
University College London
London, United Kingdom
e.letier@cs.ucl.ac.uk

M. Angela Sasse

Dept. of Computer Science
University College London
London, United Kingdom
a.sasse@cs.ucl.ac.uk

Abstract—User experience (UX) is difficult to quantify and thus more challenging to require and guarantee. It is also difficult to gauge the potential impact on users' lived experience, especially at the earlier stages of the development life cycle, particularly before hi fidelity prototypes are developed. We believe that the enrolment process is a major hurdle for e-government service adoption and badly designed processes might result in negative repercussions for both the policy maker and the different user groups involved; non-adoption and resentment are two risks that may result in low return on investment (ROI), lost political goodwill and ultimately a negative lived experience for citizens. Identity assurance requirements need to balance out the real value of the assets being secured (risk) with the user groups' acceptance thresholds (based on a continuous cost-benefit exercise factoring in cognitive and physical workload). Sentire is a persona-centric requirements framework built on and extending the Volere requirements process with UX-analytics, reusable user behavioural models and simulated user feedback through calibrated personas. In this paper we present a story on how Sentire was adopted in the development of a national public-facing e-service. Daily journaling was used throughout the project and a custom built cloud-based CASE tool was used to manage the whole process. This paper outlines our experiences and lessons learnt.

Index Terms—Requirements engineering for user experience, usability, enrolment process, Personas, Sentire, Volere, industry and research collaboration

I. INTRODUCTION

A number of user centric tools, techniques and frameworks have been proposed for the specification, design and development of e-government services, however such services “*continue to demonstrate poor usability*” while providing a “*less-than-satisfying UX*” [1]. ISO (International Standards Organization) defines UX as “*a person's perceptions and responses that result from the use or anticipated use of a product, system or service*” [21]. We argue that this is an explicit result of a number of conflicting and competing goals, arising from governments, citizens and contractors alike where reconciliation requires an objective effort where each party is given a voice from the earlier stages of a system's lifecycle. Although usability and UX are seeing their way in government projects, a study conducted across 38 project contracts for public software systems [2] Lehtonen et al noticed that in most cases statements about usability were vague and/or focused on precise

design or process features (e.g. *software must be tested for usability* and *screen must have an exit button*). Typically policy makers also specified that to ensure usability user representatives shall give a subjective rating following system demonstrations given by the contractors. Jokela and Buie [1] argue that these are not proper usability requirements since they lack verifiability, validity and comprehensiveness. Subsequently these are comparable to wish lists or strategic desires that may or may not support the user's primary task. We believe that a unified and systematic government-wide requirements framework that puts the user at the centre of the equation is missing. This must also embrace knowledge accumulation and re-use and thus reducing dependence on contractors' knowledge and experience. Such dependence can pose serious risks to project success [1]. Janowski, Estevez and Ojo [3] state that the lack of methodologies and models, as well as the lack of cohesion between related e-government projects are two of the major causes of e-government project failure. User eXperience is very difficult to quantify and thus specify in measurable terms within the requirements document or call for tenders. For this reason UX may find itself on the back-burner and contractors may de-prioritize it so as to increase competitiveness in terms of costs and time to deliver [1].

Through our Volere-inspired requirements elaboration and design framework – Sentire – we are studying ways to express the impact that enrolment-related e-service requirements can have on users (at requirements stage) in terms of perceived workload and willingness to adopt. We decided to focus on the design of acceptable enrolment processes since we believe that this is a critical success factor within the e-government domain. France [18], UK [17, 19] and Austria [20] are some of the countries that have experienced negative repercussions due to sub-optimal identity related processes. One of our main goals is to provide a framework and corresponding tools that would help designers and service providers (in this case government entities) design better e-services. We define the term ‘better e-services’ as systems that are human-centric and conducive to goal achievement, both from a citizens' perspective, but also from a service provider's point of view. This framework positions users at the centre of the requirements process and adopts calibrated personas (classical personas embellished with re-usable statistical behavioural models) to generate simulated

user feedback on critical design decisions (e.g. enrolment processes). This simulated user feedback acts as an immediate and objective starting point for the design team to assess the impact that certain design requirements might have on various user groups – enabling an iterative process of progressive improvement. This mitigates against bad design decisions being identified at a later stage (prototyping) making it more difficult and expensive to fix, or at worst, not being captured at all during user acceptance testing. The persona calibration process is based on a systematic process of understanding which also provides interesting insights on our assumption personas thus informing persona development and evolution. Sentire was used in previous case studies and interesting insights emerged [7, 10, 13]. We confirmed the mechanics in the earlier studies and validated results in subsequent ones. However we wanted to assess the latest iteration of Sentire within a fully-fledged e-government service project, from inception to launch. This will allow us to focus on the overall process and determine the impact that Sentire affords at each stage.

This paper is structured as follows. Section II provides some background on the theory behind this work (Volere and Sentire respectively) as well as project specific information including our main collaborator, the e-service to be developed, our research goals and techniques adopted. We then present the process we followed to build this e-service in Section III including our own reflections at each stage. Finally we present our conclusions and recommendations in Section IV, followed by some future work in Section V.

A. Companion Resources

This document is as self-explanatory as possible however the following resources offer a gentle introduction to the central techniques discussed throughout.

- Sentire poster: <http://goo.gl/Giy4ar>
- Sentire explainer video (basic): <http://goo.gl/9OjPKM>
- Explainer video (advanced): <http://goo.gl/x4pACJ>

II. BACKGROUND

A. What is Volere?

Volere is considered to be a practical requirements framework recommending usable notation, techniques and deliverables within a systematic and rigorous requirements elaboration process. This can be contrasted with the more formal requirements elaboration techniques, such as *i**, which may be too complex for non-technical stakeholders to follow and comprehend [14]. Volere encapsulates industry best practices for requirements elaboration and specification encouraging clarity, explicit scoping, simplicity, rigour (through specific milestones and templated deliverables), scalability, traceability, validation and testability (adopting *Fit Criteria*), re-use and a socio-technical perspective within a repeatable process [4, 14]. A complete discussion on Volere is provided in [4].

From our experience, Volere has been received positively by both policy makers and IT specialists, specifically for its rigour, process clarity and simplicity.

B. What is Sentire?

The name Sentire (“to listen”) is inspired from its underlying process Volere (“to want”) and reflects the idea of listening to end users through simulated feedback generated computationally using statistical user behavioural models: *calibrated personas*. This feedback is generated at the earliest stages of the requirements/design process. This technique does not replace traditional UX evaluation techniques (e.g. user walk-throughs, focus groups and eye-tracking studies) but provides an early-stage user-centred discipline to inform decision making on critical design aspects. Generally actual users are introduced to the process at a later stage, especially when a hi-fidelity prototype is developed. Sentire bridges this gap, introducing the users’ reactions to critical design aspects, potentially from day one. This a) provides a low-cost and immediate feedback mechanism, b) reduces the risk of major re-work on critical aspects and c) instils systematic user-centricity as part of the requirements and design process where every decision taken will result in some form of impact on the users’ experience, represented as a vector – containing the magnitude of such impact and also its direction (negative vs. positive). In line with Nielsen’s [5] task success rate measurements, Sentire provides indications of the probability that a given user group would be willing to complete the task (success). At the same time Sentire also provides NASA-TLX-based workload measurements [6] as well as other metrics acting as meta-information to explain the given indications (adopting NASA-TLX’s multi-dimensional nature). Sentire allows policy makers to specify UX related requirements such as task success rates (e.g. 80% should be able, and willing, to enrol on our service with a 95% confidence interval) and workload (e.g. perceived workload should be kept below 40% for all user groups). In collaboration with the contractor, the project team would then work towards reaching the desired goals, from day one. Sentire extends the Volere process as shown in Figure 1. In Section III we will be explaining each extension as well as their practical implications.

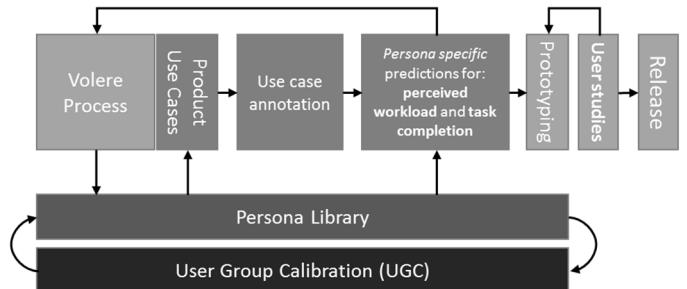


Fig. 1. Sentire requirements and design framework – extending the Volere process to enable user feedback simulations at the requirements stage

The persona library holds statistical user models that explain user behaviour when facing critical design factors. For this study we are modelling perceived workload and willingness to complete a task when faced with different enrolment processes. These user models are based on a persona calibration exercise which captures user behaviour while performing a series of pre-determined tasks. This information is then processed in a statistical package to fit two regression models (linear regression model for workload data and binary logistic regression model to

predict the users' willingness to adopt the service, explained by the probability that a user will complete the enrolment process). These models provide regression coefficients which explain the user group's attitudes towards the different design factors behind enrolment processes. These coefficients could then be used to understand the potential impact of different design alternatives on users (in terms of willingness to adopt the service and perceived workload). The tasks presented during calibration are configured differently based on design elements identified in a previous empirical exercise [7, 10] (i.e. *number of fields to fill, possible delay severity, interruptions to daily routines, number of new credentials* as well as *level of service compulsion and frequency of use*). User models are then plugged into the respective traditional persona construct representing similar users as the ones used to generate the user models – termed as *calibrated personas*. As a case in point, a behavioural model for undergraduate students can be re-used with various personas sharing the same set of demographics. Product use cases (i.e. those involving an enrolment process) are annotated with values representing enrolment specific design elements, and are also associated with active actors (represented by calibrated personas). This allows the design team to generate user experience simulations at the requirements stages, before any prototypes are built.

In order to facilitate this otherwise laborious process, a CASE tool was developed to assist designers in the aforementioned steps. The tool facilitates the generation of deliverables compliant with the Volere requirements templates. This provides an overarching structure which is highly desirable especially at the initial stages of the development lifecycle. The CASE tool is also supplemented with an online calibration portal. These techniques and tools will be discussed in Section III, however more in-depth information is provided in [7, 10, 13].

The following tools and techniques are typically adopted in Sentire-driven projects: *Sentire's CASE tool* [7, 13], *calibrated personas* [10] which in turn build upon Cooper's *personas* [8] and Faily and Fléchais' *persona cases* [9], *participatory design* [12], *user walkthroughs, card sorting* [11] and *eye gazing tests* [16]. All of this is organized within *Volere's requirements process* [4], including its templates and deliverables.

C. The E-Service

The Malta Competition and Consumer Affairs Authority (MCCA) are aiming to improve the way citizens interact with the authority while streamlining internal processes for increased efficiency. For this reason a series of meetings were carried out with top management to discuss their needs as well as our objectives, and eventually a collaborative agreement was formalised for the development of the Consumer Advice Portal (CAP) - a public facing e-services that acts as the first point of contact for consumers; offering an advice and complaints wizard, a publication repository, frequently asked questions as well as a knowledgebase on past cases. This e-service will also act as an internal knowledgebase enabling easier access to information and improve knowledge transfer and re-use for current and future case-officers. Based on this we agreed to adopt Sentire to develop CAP's requirements as well as design

and build the actual portal using an agile development methodology (categorised as a *Rabbit* project in Volere terminology).

D. Our Goals

We believe that the enrolment process is the first hurdle in any e-government project and it can make or break a service. Users may opt not to use a service simply because the workload involved to enrol exceeds their acceptability thresholds. Sentire was specifically built to inform the design process through simulated user reactions given specific service alternatives. This case study allows us to reflect on and evaluate Sentire and its associated toolset within a fully-fledged national e-government project.

III. THE PROCESS

Figure 2 outlines the workflow adopted for this project, broken down by work unit. Each work unit will be tackled in the following discussion, highlighting implications arising from the adoption of Sentire.

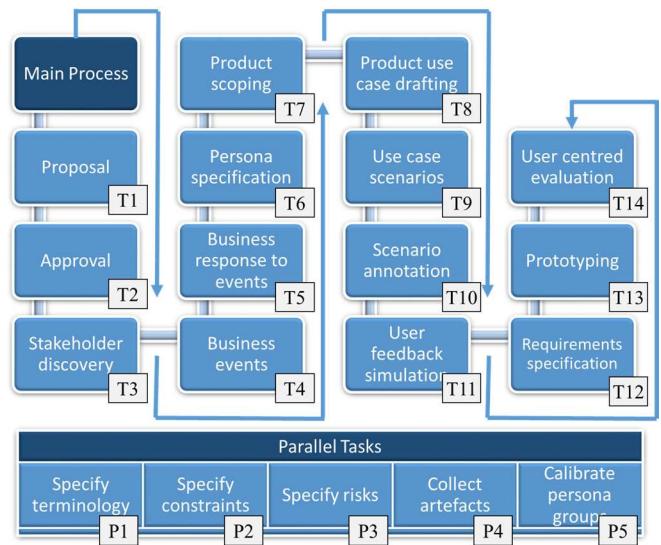


Fig. 2. Consumer Advice Portal project workflow. *T* represents sequential tasks while *P* indicates parallel tasks

In the initial stages (**T1, T2**) high level goals were specified and approved by top-management. A project team was also assigned. At this stage we started discussing entities that may have a direct or indirect influence on the e-service (**T3**). For this purpose we adopted a simplified version of Ian Alexander's onion-ring model for stakeholder identification [15]. Several pre-determined categories (from the Volere template) were also used to inform the discovery of potential people/entities/systems that can have an impact or are impacted by the authority's work. By the end of the requirements process around 47 stakeholders were discovered together with their inter-relationships (where applicable) and these included domain experts, internal specialists and other government entities.

A discussion on business events was initiated (**T4**). These are events that occur at the authority, including externally initiated events (e.g. consumer calls the help-desk) and time-triggered

events (e.g. bi-weekly batch processing of complaints). We feel that by creating an outline of business events we would be preparing a solid foundation to a) rigorously identify business use cases (how the authority responds to business events) and b) discover new areas that may need further investigation. These are foundational steps enabling a fine grained scoping exercise for the system-to-be. By the end of the requirements process 12 externally initiated events were identified along with six time triggered events (18 in all). At this point we started to understand how the authority operates and how different internal stakeholders respond to both external and time triggered events. This helped in formulating a first set of technology-agnostic business use cases (**T5**) explaining the various work processes and exceptions. Relevant artefacts, such as physical forms, documents, leaflets, meeting recordings, transcripts, photos and correspondence were stored within the project's workspace in Sentire's CASE tool (**P4**). So far all discussions revolved around the authority's work and there was no mention of any e-service related features. Initially we wanted to draw a picture of the authority's world and work as well as any form of interaction with consumers and traders. This provides more knowledge and data into which we can ground well-scoped decisions, rather than merely coming up with requirements for unknown problem/s existing in an undefined domain. Scoping becomes much easier and robust when based on broad knowledge. At this point, workflows started to emerge and various (sometimes unexpected) domain experts were involved in order to walk us through and explain the various scenarios. Business use cases were specified within the CASE tool reflecting how the authority responds to the various events. Information held included use case pre-conditions, business rules and outcomes as well as normal, alternative, exception, misuse and negative use case scenarios. Active and interested stakeholders were associated with use cases. Several domain specific terms and acronyms started to emerge and these were recorded in the terminology library within Sentire's CASE tool (**P1**). The terminology library facilitates communication across the entire team (including future team members) by providing a common dictionary of domain and project specific terms. Project constraints (**P2**) and risks (**P3**) were discussed in tasks **T1** and **T2** and also stored in their respective repositories. Furthermore, identified risks can also be linked to other aspects of the project (e.g. Use Cases) that can help to mitigate their occurrence. An online, centralized and structured repository makes it easier for team-members to monitor progress without the need to request and go through any ancillary documentation.

Following the exploration of business events and associated responses we have now formulated a clear idea of the main citizens groups who interact with the authority. A set of assumption personas was outlined while flagging those who might eventually make use of the consumer advice portal (**T6**). At this point a parallel task was spawned – persona elaboration, grounding and calibration (**P5**). Assumption personas were stored in the CASE tool and this encouraged us to collect more empirical data that would shed more light on the authority's 'clients', including actual case data and statistics. This enabled us to evolve our assumption personas even further while

grounding aspects such as user activities, aptitudes, attitudes, motivations and skills in empirical evidence. Personas were elaborated even further during the persona calibration exercise. This task ran in parallel with the main project requirements workflow.



Fig. 3. Persona library in Sentire's CASE tool

Initially four primary and four secondary personas were specified:

- **Mary Piscopo** – 55-65 year old tech-newbie
 - **Joel Caruana** – 30-40 year old teacher and mother
 - **Noel Caruana** – 30-40 year old engineer
 - **Shanya Borg** – 16-18 year old student
- Secondary persona (won't be directly affected)
- **Joe Grech** – 55-65 year old trader (not a target user)
 - **Sylvester Camenzuli** – 70+ year old retired policeman
 - **Joanne Bonnici** – 18-25 year old content manager
 - **Joseph Zammit Borda** – 30-40 year old case-officer

A number of user models have been previously generated in other case studies (for similar personas to those shown in bold) and thus can be reused. No models were available for *Mary Piscopo*. We organized a number of user group calibration (UGC) sessions with participants falling under this user archetype. Seven individuals accepted to participate. All ethical considerations recommended by UCL's Research Ethics Committee were taken. Calibration sessions took around 50 minutes to complete.

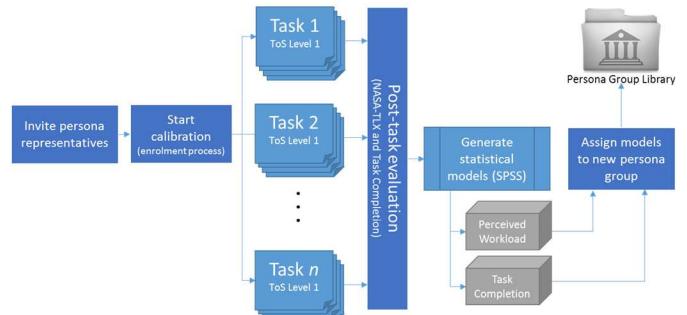


Fig. 4. Sentire's persona group calibration process

The calibration process requires participants to go through a set of nine fictitious enrolment tasks covering a wide spectrum of real-world e-service enrolment configurations, ranging from

easy (e.g. requiring users to fill in two fields and create a password) to hard (e.g. requiring users to travel physically to a government office to verify their identity as well as creating multiple credentials). After each task participants had to rate six sources of workload (*mental, physical and temporal demand, own performance, frustration and effort*) and specify their willingness to complete the enrolment process in four different e-service scenarios (i.e. services of increasing importance/compulsion and frequency of use). The online and self-administered version of the calibration process was not practical with this group of people mainly due to a low level of confidence in using online services for the first time. We devised a new approach whereby the facilitator explained each task using visual and verbal cues while calibration feedback was recorded on their behalf. This approach might pose a risk to validity due to potential bias (as opposed to going through the calibration process independently), however we noted a large degree of openness and honesty in their feedback which re-assured us on the authenticity and quality of the data generated. Leading questions were also avoided to mitigate this risk. This technique also helped to uncover several unexpected insights on this user group's attitudes towards enrolment. The calibration data from each session was consolidated and prepared for processing using a statistical package (SPSS). Two regression models were created: a *linear regression model* to explain perceived workload and a *binary logistic regression model* to predict the users' willingness to adopt the e-service.



| | Willingness to Use Coefficients | Perceived Workload Coefficients |
|-------------------|---------------------------------|---------------------------------|
| B-Coefficient | 5.866 | 3.888 |
| New items | -0.78 | No impact |
| Items to recall | No impact | 2.183 |
| Delays | -1.434 | 34.332 |
| Interruption | -1.925 | 24.127 |
| Type of Service 1 | -2.339 | No impact |
| Type of Service 2 | -1.448 | No impact |
| Type of Service 3 | -0.718 | No impact |
| Type of Service 4 | No impact | No impact |

Fig. 5. These coefficients explain a user group's attitudes towards enrolment specific design elements (e.g. an additional form field increases perceived workload by 2.2%, while a delay increases perceived workload by 34.3% while reducing the probability of adoption by 1.4%). Statistical tests are conducted on each model to assess its predictive power and validity – calling for further evaluation and possibly calibration

Throughout the process we noticed that some participants behaved in a significantly different manner, even though they were theoretically accurate representatives of our persona. This led us to believe that our initial assumption persona (*Mary Piscopo*) may have been an over-generalization of that particular user group and we considered this as a sign that new personas might be emerging.

We started to generate more insights on our participants and a new set of attitudes started to emerge clearly suggesting that we're dealing with two different user groups. This new group of people appeared to be extremely afraid of "*breaking something*" even though they are willing to adopt new technologies. Another attitude was that they prefer physical demand (i.e. going somewhere) than feeling frustrated (i.e. fear of doing something wrong or lack of understanding). This was related to the issue of confidence, specifically wanting to get the job done with high confidence in the outcome.

We decided to create a new persona and associated user models to cater for this variant on the original persona (we called this variant *Doris Piscopo*), based on these new insights.

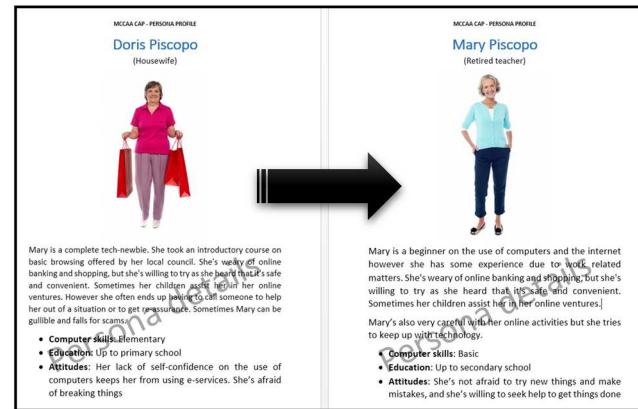


Fig. 6 Creation of new persona to reflect an emerging user archetype

A qualitative analysis of the calibration sessions confirmed our hypothesis for this new persona. The following sum this up:

- "When I see an enrolment page I stop as I'm afraid of breaking something [the computer]"
- "I depend on my daughter who's still at home ... when she leaves, then I'll make an effort to overcome my fear"
- "I prefer to go out and finish tasks in person, it's part of life and it's relaxing"
- "Time and physical effort are not an issue"

"Lack of self-confidence" and "fear of breaking things" are two main attitudes associated with this new persona. These observations are grounded in feedback obtained throughout the calibration process. In the final part of the calibration process (comparison of NASA-TLX workload dimensions) it became extremely clear that this group of people are more willing to endure *physical* and *temporal demand* as long as they feel confident that they completed the job, they did it "the right way" and that they're not making any mistake that could cause harm or damage anyone or anything. One participant noted that she has a "*fear of breaking the national network [laughing]*". The calibration exercise helped us learn more about our users and

their attitudes towards specific design factors, while enabling us to operationalize and model them for future re-use.

In the meantime tasks **T3**, **T4**, **T5** as well as **P1** provided us with a solid foundation to scope the new e-service (**T7**) while evaluating possible product use cases, or rather functions that the new e-service may afford to stakeholders, primarily consumers (**T8**). Following Volere's templates, each product use case was discussed and scenarios (normal and alternative) were drafted (**T9**). All of the personas were present in a visible location during these sessions.



Fig. 7. Persona posters were highly visible during meetings

Given the required agility for this project, a hi-fidelity prototype was used to guide the team in the requirements specification and design process. We believe that in projects where high agility is required, product use case and scenario development are in themselves a design activity, rather than just requirement elicitation and specification techniques. As part of the product use case specification exercise we conducted a card-sorting exercise to determine an initial take on the e-service's information architecture. A set of yellow post-it notes representing functional and informational pages were provided together with a set of blank green ones. The group was asked to use the green notes to create categories under which the other post-it notes would be placed. This would in turn translate into a consensus based hierarchical representation of information and service pages. Following a team effort and based on experience and expectations the team managed to come up with new informational and functional categories for the e-service, while existing ones were removed or consolidated. The card-sorting exercise was crucial to reflect on specific requirements: *how will a user look for advice? Will consumers follow a hierarchical navigation pattern through FAQ drill-downs?* This shaped our idea of flows and an initial design started to emerge. Participants also had to agree on naming conventions for groups of concepts, and this informed the creation of menus, menu items and layouts (e.g. "News" vs "Alerts"). We planned to use closed card sorting (with a predetermined set of categories) however the team opted to create new categories and concepts where necessary. This gave way to discussions about user-journeys, defining the path a user would take to complete a task (e.g. seek advice or file a complaint). Using a marker the team joined various post-it notes with arrows, indicating flow. At each point throughout the discussion we referred back to our personas to determine whether we are excluding any specific user groups. The guiding principle was to design for the lowest-denominator in terms of skills. Whenever a design decision was taken we referred back to our personas and asked questions such as: "*Would it be*

intuitive to Joe? Would it be too complex to Mary? How would Shanya react to this?" This was however based on subjective interpretation. Following Faily and Fléchais' [9] recommendations we attempted to back our assumption personas with facts about real consumers who have interacted with the authority. Grounding was also based on first-hand experience by stakeholders.



Fig. 8. Left: card sorting exercise following an initial iteration of product use case designs. Right: second stage of the card-sorting exercise was to determine user journeys

User journeys convey important information to the design team as they uncover flaws in assumed workflows and allow for an early rethink of the steps required to complete a primary task. This exercise should be conducted with both users' goals as well as the authority's goals in mind. At this point we had a set of product use cases, built following a systematic investigation of the authority's work and a detailed scoping exercise. Certain requirements started to emerge, and a note was taken (although no formal specifications were made at this point). We proceeded to test the current product use cases to determine their impact on users (**T10**, **T11**). Various advantages and disadvantages of introducing enrolment were discussed, both from a users' point of view but also from MCCA's point of view. Two enrolment approaches were considered; either via the national e-ID infrastructure or via a custom-built and internal user management facility. These scenarios placed different demands on users, and the impact from each scenario was assessed individually. From a users' perspective we considered physical workload (i.e. national e-ID requires users to visit a registration office in person) and cognitive workload (i.e. custom MCCA user accounts would require users to create yet another set of credentials). Sentire was adopted to simulate and visualize the impact that enrolment can have on users. Normal case scenarios were created for both enrolment options and these were then annotated (**T10**) as shown in Figure 9. Each use case is specified using scenarios, which are in turn specified as consecutive steps. Enrolment-specific steps are annotated with measurements for each of the design factors used in calibration. This will allow us to generate predictions using our statistical user models.

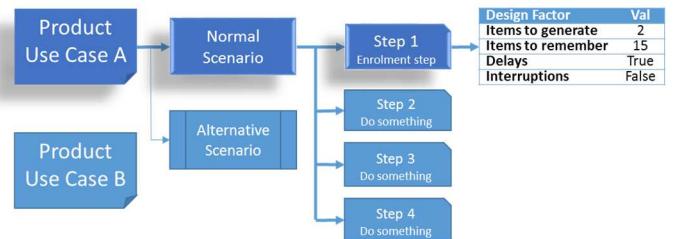


Fig. 9. Sentire's product use case annotation workflow

We then moved on to simulate user feedback through Sentire for the several user groups involved and for the different enrolment scenarios (**T11**). Before computing simulations the available user models were associated to the project's primary personas as shown in Figure 10.

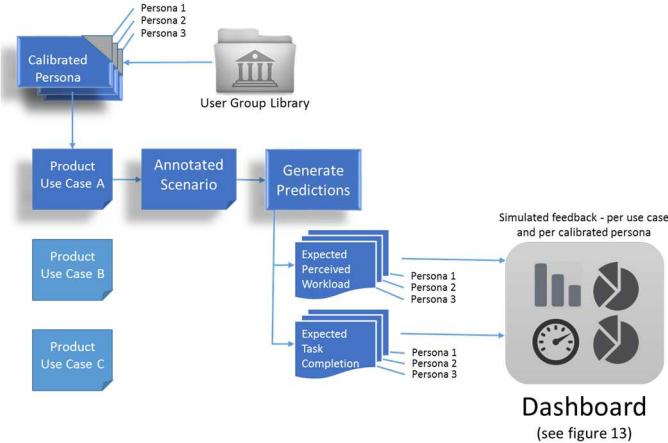


Fig. 10. Sentire's user feedback simulation workflow

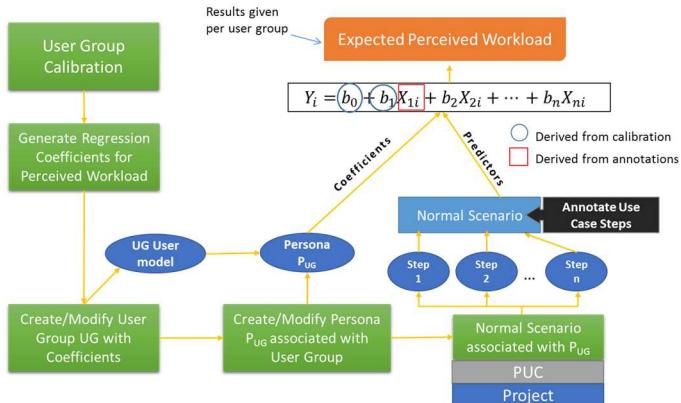


Fig. 11. Sentire's simulation algorithm for perceived workload (willingness to complete the task is calculated using a binary logistic regression function as shown in Figure 12)

$$P(WtCT) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + \dots + b_n X_n)}}$$

Number of users (per persona) who would willingly complete the task
 $= P(\text{Success}) \times \text{Persona population}$

Fig. 12. Willingness to complete the task (WtCT) uses a binary logistic regression function

Personas are associated with statistical user models generated through calibration. These models denote regression coefficients that explain the users' reaction towards specific design elements (e.g. an additional form field will add 10% to the perceived workload and reduces the chance of task completion by 3%). These values, together with the product use case measurements (annotations) are parameterized into their respective regression functions. This provides us with predictions for perceived workload levels and willingness to complete the primary task (as opposed to giving up at enrolment stage). The simulated feedback confirmed the team's concerns and also strengthened their conviction that adopting an enrolment process based on the national e-ID for basic and

infrequent transactions is an overkill, which would then result in major adoption issues. We were operating on the assumption that the number of active e-ID accounts in Malta is relatively low (i.e. around 10%). Using an MCCA account would result in generally improved process completion rates (sign-ups) however this has a negative impact on users represented by *Mary* and *Doris Piscopo* while discouraging many younger users represented by *Shanya Borg* (16-18 year old users). Sentire confirmed the team's gut feelings and decisions could be taken with higher confidence, backed up with objective data.

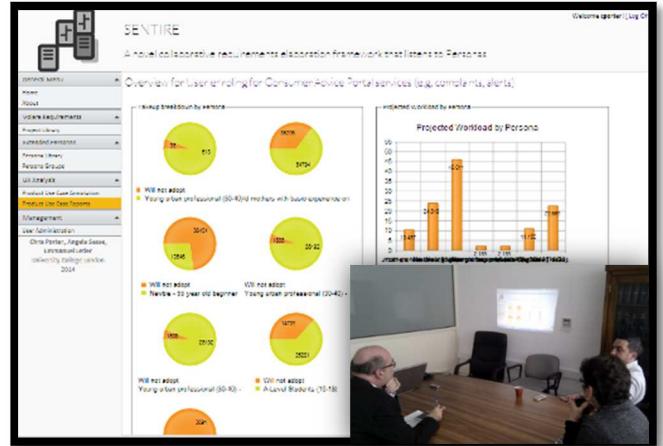


Fig. 13. Simulated feedback for the different user groups represented by the various calibrated personas used throughout the design process. The feedback shown above was generated using our Sentire CASE tool on the “enrol with an MCCA account” product use case (Inset: Sentire workshop participants)

For each group of users Sentire reported (see Figure 13) the amount of perceived workload (*histogram*) as well as the willingness to enrol (*pie-chart*) for the e-service and complete the task online (as opposed to using conventional methods to file complaints and seek advice via phone, email, snail-mail or in person – or give up on the process altogether). Following a couple of iterations (changing enrolment process parameters) it was decided to leave the e-service open across all use cases without the need for any compulsory enrolment. Nonetheless an optional MCCA account was considered to be a reasonable offering for those consumers who would wish to track their complaints and other interactions.

Persona calibration lets us capture, model and store user behaviour which can then be re-used via Sentire's persona library. Although not necessarily precise, simulated feedback challenges designers to re-assess their assumptions and decisions following a systematic and repeatable process. At this point a set of product use cases were formulated and tested for critical enrolment-related issues. Atomic requirements were specified for these use cases adopting a modified version of Volere's requirements Snow Card template (**T12**).

These low-level requirements covered various categories including *functional*, *usability and humanity*, *look and feel* and *maintainability and support*, all of which are testable, measurable and traceable. Each requirement was assigned to specific *Product Use Case/s* or marked as *Global*. This offers various levels of requirements granularity, with the product use

case being a high level view of what the system shall do, and atomic requirements specifying low level detail denoting how a system shall achieve such functionality.

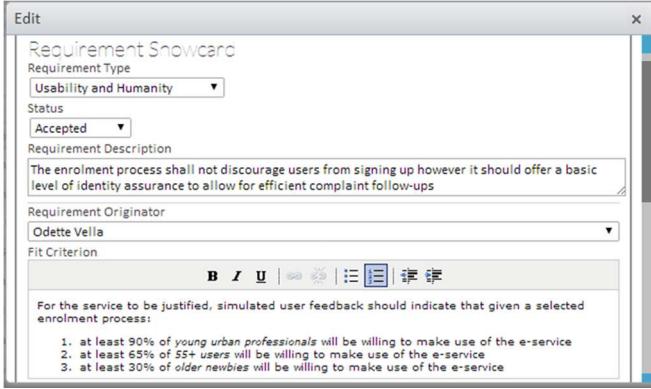


Fig. 14. Atomic requirements were specified for these use cases adopting a modified version of Volere's requirements Snow Card template (and using measurable UX-related fit-criteria).

Sentire's CASE tool offers a project visualization map, allowing the team to view requirements at various levels of granularity as well as their inter-relationship with higher level groupings (e.g. product use cases, business use cases and events). Other project elements are also displayed (and linked), including stakeholders, personas/actors, events, risks, use cases and requirements. This can serve as a visual impact assessment utility for regression testing following modifications to requirements or use cases.

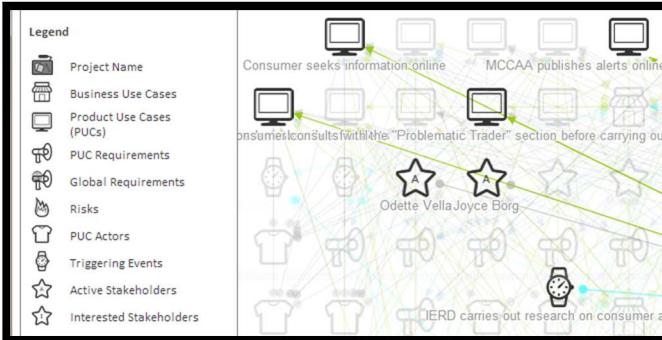


Fig. 15. Sentire's CASE tool: project map – indicating relationships between different elements within a project.

Based on the current level of coverage and detail, a prototype started to emerge (**T13**). The hi-fidelity prototype was tested with a number of participants at University of Malta's Usability Lab (**T14**). Using Tobii's eye-tracking and analysis studio, a set of pre-determined tasks (goals) are provided during which eye-gaze data is captured for a deeper assessment on findability, navigability and explicit pain-points (e.g. using heat-maps to uncover points of failure). Retrospective Think Aloud (RTA) sessions provide us with deeper and invaluable insights and knowledge on what users expect, what they look for and the rationale behind their decisions. This data supplements the eye-tracking information.

A number of severe usability issues were uncovered during the first few sessions. These were corrected prior to the

subsequent sessions. Following this iterative process the authority can now plan case officers' training, data migration and release.



Fig. 16. Prototyping the e-service based on the initial information architecture session



Fig. 17. Left: Tobii X-120 was used for the eye-tracking sessions. Right: Participant during an eye-tracking session



Fig. 18. Heat-maps indicate common gaze patterns across individuals

IV. CONCLUSIONS AND RECOMMENDATIONS

A. Persona Evolution through Calibration

An interesting side-effect of the persona calibration exercise was the identification of new personas stemming from behavioural information. Our primary persona *Mary Piscopo* was built on assumptions, primary data, observations, opinions and experience provided by the stakeholders. When representatives of this persona were calibrated it was immediately clear that there were several distinctive behavioural aspects between various participants who were theoretically similar in demographic terms. This informed the evolution of *Mary Piscopo* but more importantly to the creation of a new persona *Doris Piscopo* which reflected the contrasting attitudes towards specific enrolment-related design factors. Although a similar result could have emerged through ethnographic methods we believe that the systematic, structured and objective nature of the calibration process highlighted marked differences

between expected and actual behaviour across participants. When distinct clusters of behavioural information emerge from the data then this would be indicative of a phenomenon that requires further investigation (e.g. out of seven participants, three generated commonly divergent results with respect to perceived workload and willingness to complete a task). A small number of participants may not yield enough information to generate statistically significant results (models), however these are all important insights that could drive the team to consider different angles of the same problem while being critical of their own actions and conclusions - promoting a structured and actively reflective design process. An unstructured and qualitative discussion with participants might not have the same effect as the systematic and repeatable user group calibration (UGC) process.

B. Partial User Models

Sometimes user models can only partially capture user behaviour and reactions towards specific design factors. This is mainly because certain predictors used during the calibration process would not be statistically significant for a specific group of users (e.g. for a given user group and in a given context, an increasing amount of fields to fill may have an impact on workload but not on the willingness to complete the task). Another major cause of partial user models is when data collection is not extensive enough to perform good model fitting (using linear or logistic regression) and the influence exerted by certain predictors would degrade the model's overall predictive power. Statistical tests would highlight these predictors which are in turn excluded from the final model to a) strengthen the effect of the remaining predictors and b) improve overall predictive power, albeit for specific design factors only. This however must be made explicitly clear to avoid misinterpretation. Figure 19 shows a conceptual representation of how this can be communicated.

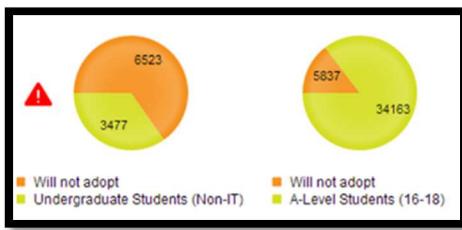


Fig. 19. A conceptual representation of alerts shown when simulations are generated via partial user models (i.e. first chart is based on a partial model).

C. Indicative can be as Good as Precise – as long as it's comparable

The team described the experience as intuitive and non-threatening to the non-technical person. This is mainly due to its technology-agnostic Volere-based process. Simulated feedback provides a truly user-centric and objective grounding for discussion and from the experience gained throughout this particular project it was clear that the team was not interested in precise predictions but considered general trends or indications to be sufficiently useful to inform decision making. Management felt more comfortable taking decisions based on a quantitative (and comparable) view of the impact that certain design

decisions would have on users. The introduction of a new calibrated persona (*Doris Piscopo*) was a case in point where its behavioural simulations were only indicative and based on a weak statistical user model. This was mainly because data for the underlying user group (*complete newbies (55+)*) was based on readings from only three UGC participants. Nonetheless this calibrated persona was still useful as it gave an objective, yet broad indication of what kind of reaction to expect in comparison to the other calibrated personas. Additional calibration sessions would strengthen the user model underlying this persona. Nonetheless the current, albeit weak model still contributed towards a design decision, that of eliminating all enrolment processes from the e-service.

D. Contextual Feedback is Possible with In-Context Calibration

Persona group calibration can be carried out either in a lab environment or within the users' natural environment from where the e-service will be used. In an earlier case study we calibrated participants representing the *young urban professional (30-40)* user group. In this case calibration was carried out at the participants' workplace and it was noted that decisions were highly influenced by contextual nuances. Participants could refer to their physical surroundings and work-conditions before submitting their feedback on perceived workload and task completion (e.g. "how hard would it be to scan a utility bill over here?"). Behaviours and attitudes might change in different contexts, however in-context calibration may mitigate this risk by creating behavioural models influenced by contextual nuances.

E. Re-use

Knowledge gained throughout the process, including terminology, stakeholders, personas, behavioural models and requirements have been stored in the online CASE tool and may be re-used for future projects by MCCA or possibly by other national and/or regional entities. This reduces the *cold-start problem* in future projects since accumulated knowledge would be available for immediate re-use – potentially resulting in less time spent on aspects such as user group calibration. Some critics have argued that the expense to calibrate personas could be avoided and resources should be used to test e-services directly with users. We believe that Sentire is not intended to replace user testing but its purpose is to assist design teams from the earliest stages of an e-services' life-cycle (through simulated UX-analytics) to mitigate against bad design decisions on critical success factors (e.g. enrolment). A systematic method is provided to build a grounded understanding on different user groups. This contributes towards a persona-based decision support system integrated within an industry strength requirements process. We strongly believe that Sentire's value increases as more user-specific knowledge is accumulated and maintained for use within future e-services.

V. FUTURE WORK - TOOLSET

Stakeholders at MCCA were too busy to actively engage using the online CASE tool. This required email-based updates and calls for feedback. All correspondence was stored in the

tool's project repository (as HTML documents). We believe that *multi-channel alerts* would encourage engagement. Stakeholders could be given the option to select their preferred way/s to receive alerts in case any element within the project is modified (e.g. SMS notifications when specific requirements are modified and email notifications for changes in the terminology section).

Meeting notes were taken in an online notepad made available in the project workspace, however following a number of meetings this got too large and impractical to use for reference purposes. A project *meeting management area* could be introduced – offering meeting recording facilities (meeting notes and meta-information) as well as a meeting-specific repository for artefacts collected during each meeting (e.g. photos, recordings, and documents).

Use case annotation and feedback simulation should be made available on a *sidebar* within the use case formulation page. At the moment user feedback simulations are only available from a separate area within the tool and thus creating a break in the workflow. A realtime *chat* facility would offer a centralized space for teamwork. Some stakeholders prefer using email, and an additional feature could allow users to send *emails directly* into Sentire's CASE tool by providing project-specific addresses, such as mccaa_project@devbell.com. Wireframes are currently created using external tools and images are then imported into the CASE tool. We're working with a wireframing app vendor to create an *integration API to offer in-app collaborative wireframing* capabilities.

More work is also required to study the effectiveness of and to improve the calibration process for use with non-technical participants.

ACKNOWLEDGMENT

We would like to show our gratitude to the Malta Competition and Consumer Affairs Authority (MCCAA) for their time and support. We would also like to thank all the persona calibration participants for their time and patience.

REFERENCES

- [1] Jokela, T., & Buie, E, "Getting UX into the contract," in E. Buie, & D. Murray (Eds.), *Usability in government systems: User experience design for citizens and public servants* (pp. 251-266) MK – Elsevier, 2012.
- [2] Lehtonen, T., Kumpulainen, J., Jokela, T., & Liukkonen, T. Nm "To what extent usability truly matters? A study on usability requirements in call-for-tenders of software systems, issued by public authorities" in *Proceedings of NordiCHI 2010: Extending Boundaries*, Reykjavik, Iceland. pp. 719-722, 2010.
- [3] Janowski, T., Estevez, E., & Ojo, A, "A project framework for e-government" in *Proceedings of the 4th International Conference on E-Government*, Copenhagen, 2005.
- [4] Robertson, S., & Robertson, J. *Mastering the requirements process: Getting requirements right*. Addison-Wesley, 2013
- [5] Nielsen, Jakob. "Usability Metrics". Nielsen Norman Group. <http://www.nngroup.com/articles/usability-metrics/>, 2001 (accessed March 2014)
- [6] S. Hart G. and L. Staveland E., "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research," *Advances in Psychology - Human Mental Workload*, vol. 52, pp. 139-183, 1988.
- [7] C. Porter, A. Sasse M. and E. Letier, "Designing acceptable user registration processes for e-services," in *Proceedings of HCI 2012 - The 26th BCS Conference on Human Computer Interaction*, 2012.
- [8] A. Cooper, R. Reimann and D. Cronin, *About Face 3: The Essentials of Interaction Design*, Wiley, 2007.
- [9] S. Faily and I. Fléchais, "Persona Cases : A Technique for Grounding Personas," *Proceedings of the 29th International Conference on Human Factors in Computing Systems*, pp. 2267-2270, 2011.
- [10] C. Porter, M.A. Sasse and E. Letier, "Giving a voice to personas in the design of e-government identity processes," in *Research to Design: Challenges of Qualitative Data Representation and Interpretation in HCI - in BCS HCI 2013*, 2013.
- [11] Nielsen, Jakob. "Usability Testing of WWW Designs: Article by Jakob Nielsen." Nielsen Norman Group. <http://www.nngroup.com/articles/usability-testing-1995-sun-microsystems-website/>, 2001 (accessed February 25, 2014).
- [12] Trigg, Randy, and Andrew Clement. "Participatory Design." Computer Professionals for Social Responsibility. <http://cpsr.org/preBSITE/program/workplace/PD.html/>, 2000 (accessed March 6, 2014).
- [13] Porter, C, "SENTIRE / designing human-centric identity systems". Poster session presented at the Information Assurance Advisory Council (IAAC) Symposium, London, 2013.
- [14] Faily, S, "A framework for usable and secure system design", University of Oxford, 2011.
- [15] Alexander, Ian, and Suzanne Robertson. "Understanding Project Sociology by Modeling Stakeholders." Stakeholders without tears. http://www.scenarioplus.org.uk/papers/stakeholders_without_tears/stakeholders_without_tears.htm (accessed March 6, 2014).
- [16] K. Pernice and J. Nielsen, "How to Conduct Eyetracking Studies," Nielsen Norman Group. 2009.
- [17] HM Revenue & Customs, "Online services: digital certificates", retrieved May 2012, from HM Revenue & Customs: Online Services: Digital Certificates: <http://www.hmrc.gov.uk/ebu/digital-certs.htm>, 2012.
- [18] OECD, "Security and Authentication Issues in the Delivery of Electronic Services to Taxpayers". Retrieved October 2012, from <http://www.oecd.org/>, 2012.
- [19] Egovbarriers.org, "Deliverable 3: Solutions for eGovernment - Section 3: Solutions for the seven eGovernment barriers", retrieved May 2012, from Breaking Barriers to eGovernment: <http://www.egovbarriers.org/>, 2007
- [20] Martens, T, "Electronic identity management in Estonia between market and state governance" in *Identity in the Information Society*, 2010.
- [21] ISO. "ISO DIS 9241-210: Ergonomics of human-system interaction – Part 210: Human-centred design process for interactive systems". ISO, 2008.

Competition and Collaboration in Requirements Engineering: A Case Study of an Emerging Software Ecosystem

George Valen  a

Department of Statistics and Informatics
Federal Rural University of Pernambuco
Pernambuco, Brazil
Email: georgevalen  a@deinfo.ufrpe.br

Carina Alves^{1,2}, Virg  nia Heimann
Informatics Center¹
Federal University of Pernambuco
Pernambuco, Brazil
Email: {cfa, vmch}@cin.ufpe.br

Slinger Jansen, Sjaak Brinkkemper
Department of Information
and Computing Sciences²
Utrecht University
Utrecht, the Netherlands
Email: {s.jansen, s.brinkkemper}@cs.uu.nl

Abstract—Increasingly, small to medium software producing organisations are working together in collaboration networks to supply complex compositions of their products and services to customers. In this paper, we present a case study of two software companies that are evolving their partnership towards the creation of a software ecosystem. We investigate the impacts of their tightening partnership on software product management, with a focus on requirements engineering practices. We observe that the requirements definition and negotiation processes are directly affected by their fluid collaborative and competitive relationships. Power disputes, volatile roles and mismatches in release synchronisation are also aspects observed in the studied software ecosystem. We extract several observations from the case study that support small to medium software firms in making decisions within their software ecosystem.

I. INTRODUCTION

The changing dynamics of the globalised software industry have influenced companies to operate in a complex and networked setting. Software companies are increasingly redefining roles and patterns of cooperation and innovation according to their position in a value chain [1]. When competing in the same market, companies need to interact to survive in a turbulent environment. This emerging concept is called Software Ecosystem (SECO), which is defined as “a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them” [2]. Software ecosystems differ from global software engineering, where development is dispersed in several organisations that do not own the software produced and interaction among them is rather transitory. In a software ecosystem setting, a common technology or platform binds the software products. SECOs are based on the notion of organisational ecology, with a radically different business model where frequent interaction and value sharing between players promote self-regulation in the network [3].

Yu [4] suggests that for an ecosystem to operate healthy, an energy source is necessary. In a biological ecosystem, the sun is the energy source. In a software ecosystem, the market is the main source of energy that can directly and indirectly affect all players of the ecosystem. By establishing

collaborative and complementary relationships with suppliers and customers, software companies participating in a SECO can cope with financial, time and knowledge constraints [5]. Moreover, they can co-evolve in a hub of local or global market, complementing features and dividing R&D costs [6].

Software ecosystems introduce new challenges in business, social and technical levels [7]. Companies interacting in a SECO shall trigger efforts towards an increasing platformisation of their products for outside development, via extensible component-based architectures [6]. Software maintenance and evolution are important issues to be addressed in software ecosystems. For instance, players have to manage the risks of reusing components with potential low technical quality [8]. The interaction among partners also raises the need for knowledge management to support communication and efficient propagation of information [2]. Since products features are provided by different partners in an ecosystem, requirements definition and negotiation are also affected by their specific interests [9]. Software ecosystems provide products for a wide market, which means that requirements are often jointly defined by the suppliers of the ecosystem rather than elicited from users. However, actors may have conflicting priorities that have to be reconciled. For instance, individual requirements must be intertwined and merged into a set of complementary features. This increasingly networked structure requires new business models to address issues of ownership and open innovation.

Several studies have investigated software ecosystems from various domains, such as open source [10], app stores [11] and online games [12]. However, relatively little scientific knowledge is available regarding the dynamics of an emerging software ecosystem composed by Small and Medium Enterprises (SMEs). The major strength of SMEs is often their flexibility and adaptability. They are often more skilled than larger companies in adapting to new customer requirements and adopting innovative methods. On the other hand, SMEs usually have limited financial and human resources. To strengthen their market presence, growth strategies include establishing joint ventures and alliances.

In this paper, we investigate how SMEs already engaged in collaborative networks evolve their products towards a software ecosystem. We aim to understand how software product management and requirements engineering, in particular, are conducted by partner companies. From an industrial perspective, this study aims at understanding the context and challenges faced by SMEs willing to create a healthy software ecosystem. We conducted a case study of two Brazilian software suppliers that are creating an ecosystem with other partners.

This paper is structured as follows. In section 2, we present the research method adopted, which includes the research questions, data collection and analysis procedures. Section 3 presents the results from the case study. In Section 4, we discuss a set of observations that synthesise important issues faced by the studied companies in the context of their software ecosystem. We also analyse these propositions in light of related literature in the field. Finally, in Section 5 we present conclusions and future work.

II. METHOD

The focus of this case study has been to investigate how SMEs evolve their relationships to establish a software ecosystem around their products. We refined this objective in the following research questions (RQ):

- RQ1 - How is the process of software product management adjusted to cope with the challenges introduced by tighter collaboration with an ecosystem partner?
- RQ2 - What are the implications for requirements engineering in a software ecosystem composed by SMEs?
- RQ3 - How do partnerships evolve among SMEs towards the creation of a software ecosystem?

To answer the research questions, we conducted a case study of two small to medium software companies that are forming a software ecosystem. We adopted an information-oriented selection and purposively chose companies that were representative to our study on the basis of expectations about their information content. An interpretative case study supports our primarily exploratory and descriptive purpose of comprehending a particular situation through participants interpretation of their context [13].

A. Data Gathering and Analysis

Empirical data was collected through open-ended and semi-structured interviews. We defined an interview protocol with thirty-five questions that were directly refined from the research questions. This instrument ensured that the same basic structure was followed in each interview. However, new questions could be brought up according to interviewees discourse as a way to provide a more in-depth understanding of the investigated topics. Based on the protocol, we defined the sampling of interview participants. The research questions required the coverage of technical and managerial roles from both companies. Using a snowball sampling strategy, interviewees were asked to recommend other participants based on their expertise in the field.

The interviews were carried out with eleven professionals. Each interview lasted from 20 to 70 minutes. During each one, we agreed upon a nondisclosure agreement (NDA) to handle data anonymously. Participants were asked to explain company's internal documents such as requirements specification and project plans. These artefacts complemented the interviews and were considered additional sources of evidence. We further contacted participants by e-mail to clarify and extend data gathered during the interviews. Table I provides an overview of the participants.

TABLE I
PARTICIPANTS

| Company | Job Function | Years at the company |
|---------|--------------------|----------------------|
| A | Project Manager | 5 years |
| | Business Analyst | 3 years |
| | System Analyst | 5 years |
| B | Project Manager | 6 years |
| | Product Manager | 11 years |
| | Release Manager | 4 years |
| | Integ. Team Leader | 1,5 year |
| | Business Analyst | 6 years |
| | System Analyst 1 | 2 years |
| | System Analyst 2 | 3 years |
| | Tester | 3 years |

The different number of participants interviewed per company was due to the fact that professionals from Company A play several roles. For instance, the project manager also acts as product manager and product owner during requirements prioritisation. The system analyst is also responsible for software development and testing. In its turn, Company B separates these functions in specific roles, such as the release manager and the integration team leader.

All interviews were recorded and further transcribed by two researchers. With the support of Weft QDA software tool for qualitative research [14], we analysed interview transcripts through a coding procedure. We classified related quotations in labels and subsequently assembled them in categories. Then, we examined facts, concepts, ideas and their relationships. Based on these results, we derived a set of observations to explain how studied companies are forming a software ecosystem, and how the software product management and requirements engineering practices are affected.

B. Case Companies

The case study reported in this paper results from the investigation of two software organisations situated in Recife/Brazil. Company A produces software products for retail chains, distributors and wholesalers markets. It has five software products in the portfolio. The company was founded in 1986, and currently has 70 employees. In 2013, the company has achieved level F at the Brazilian Software Process Improvement Program (MPS.BR) [15]. This program is inspired on CMMI process improvement approach. In the case study, we explored software integrations around the company's main commercial product. The product is an ERP that automates business rules of distribution, wholesale and retail network,

focusing on the management of suppliers and products for resale. This product is used by more than 2.000 registered users from 12 customers in Brazil and abroad. In addition, the company provides system deployment and bespoke development services to specific customers.

Company B is specialised in developing management systems and providing consulting services. Its main product is a complete ERP solution that provides business automation for several market niches, such as: health, oil and gas, sugar industry and logistics. The product's main strategic differential is the module of accounting and tax compliance that is automatically updated with the frequent changes of Brazilian accounting standards. The company has around 300 employees distributed in its headquarter in Recife and branch offices in Maceio and São Paulo. In the last years, the company has established a leadership position within regional IT industry, which results from a portfolio of 15 software products. The company has a strong focus on software quality, having achieved the following certifications: ISO9001, CMMI level 2 and MPS.BR level C. We investigated the context of its main software product, an ERP which currently is adopted by 16.000 users from 600 customers in Brazil and abroad.

III. RESULTS

In this section we present the results obtained from our study in Companies A and B. The subsections correspond to the categories that emerged from the case study analysis.

A. Portfolio Management

The product investigated at Company A provides enterprise procurement solution for retail chains, distributors and wholesalers markets. In particular, it provides stock supply, inventory management and tax review functionalities. Company B serves the market with a full ERP focused on accounting and finance areas. Accounting parameterisation is a distinguishing feature provided by this system, which allows users to fully configure any accounting entry. In both companies, product integrations can occur in the following ways: between products from partners, with bespoke systems from customers or with products from other suppliers that serve a particular customer. Currently, the product from Company A is integrated with 15 other products, while the product from Company B has 110 product integrations. The studied companies have partnerships with local, national and international companies.

To better structure upcoming releases and increase customer satisfaction, Company A decided to focus on specific features of the product. Finance and procurement were prioritised over tax and accounting functionalities. The company modularised the system and started to establish integration partnerships with other suppliers. According to the company's project manager, "*partners are experts on features that the product does not provide or features that were discontinued*". By complementing each others functionalities, partners support the co-evolution of their products. This trend is reinforced by the product manager of Company B: "*we establish a partnership for specific parts of the system that we have*

no intention to develop". To select products and respective partners, this company defines a set of criteria that partners should satisfy. In particular, they look for partners that allow the company to enter in new vertical markets.

The companies generally initiate a partnership when facing challenges in a new market segment, as exemplified by the project manager of Company B: "*Since we do not have enough experience in the retail segment, we are having problems to cope with the flow of innovations in the field*". Therefore, Company B proposed the collaboration with Company A as a strategy to enter in the retail market. For Company A, the main motivation to establish the partnership is due to the superior accounting features of Company's B product.

The products are commercialised under different contract types. Customers may acquire the product from Company A by renting it for a fixed price during a pre-defined period, or through a project involving consultancy and a monthly fee that is annually adjusted. Company B provides two contract models: a monthly fee that is calculated according to the number of licenses rented and the customer size, or a reduced monthly fee to cover maintenance costs.

B. Product Roadmapping

Product roadmaps in the case companies typically cover one year. At Company A, the product committee defines the product roadmap by analysing potential market demands and new features to fulfill these needs. The project manager from this company noted that the product roadmap is a "*market demand to define a proactive vision of the product improvements*". In the beginning of every year, customers require the product roadmap to understand what releases are planned and how the product will evolve. In some cases, roadmaps can be developed in close collaboration with customers to create customer-specific product roadmaps. These roadmaps establish a commitment between both parties and can be used as a basis to negotiate the integration contract. The project manager added that, depending on top management decisions, information about specific features can be retained to increase product value.

The product manager at Company B develops the product roadmap aligned with the company's strategic planning and based on customers' feedbacks. According to him, to enter in a new market segment "*the product must be adapted to speak its language... Customers want to know how the product features fulfill the business processes of their segment, i.e. which features will enter in the product roadmap, in which order, at what time, etc. After I understand the domain and the products that support this segment, I start to propose new features for the product*".

Currently, Company B faces challenges in balancing resources to develop customer-specific customisations and ensure the evolution of the product according to the planned roadmap. To address this situation, the development teams have been assigned to dedicate more time developing new features from the roadmap than satisfying specific demands from customers. This prioritisation has been aligned with the

company's strategic plans to target new markets. This means that the product evolution aims to innovate the platform rather than adding customer-specific features into the product.

C. Release Planning

Both companies define the release planning aligned with their product roadmap. At Company A, the project manager and members of the development team form a product committee to plan the releases. At Company B, the product manager is responsible for the definition of next features to be launched. She arranges meetings with the business analyst to detail each new feature and provide development guidance for developers. Interviewees stressed that quite frequently emergent demands from customers can change the release planning.

Product releases are of two types in the case companies. Regular releases can address legal issues and provide novel functionalities. These new features are part of the product roadmap or originate from customer-specific customisations that companies consider relevant to include in the standard version of the products. In addition, frequent bug fix releases are detailed in release notes sent to all customers. At Company A, there is a regular product release every six months. Release frequency in Company B has been gradually increased in recent years. Latest versions were launched every month to speed time-to-market. However, this strategy created several problems:

- The short release cycle increased the number of bugs in the product. It was required to release several intermediate versions to fix bugs;
- Customers considered the steady flow of new versions disturbing, since updating the product frequently may represent a risk for their running systems;
- The company adopted a reactive approach by prioritising customer-specific customisations and neglecting the implementation of new features from the product roadmap;
- The very short release schedules impacted the routine of team members, who were under strong pressure to package new releases every month. Since team sizes remained the same, the company was demanding unrealistic efficiency from the teams.

After recognising all the issues caused by the short release cycles, Company B started to launch new product releases every two months. The current goal is to adopt a biannual periodicity for regular releases and launch additional versions for specific customers every two months. During integration projects, partner companies must deploy their products simultaneously at customer site. However, release planning cycles are not properly synchronised and product versions are launched in different moments. Development teams from both companies lack a centralised communication and they are not coordinated to treat integration conflicts. In particular, although a project manager from a given partner may be responsible for the integration project, he has no authority over the development team of the other company. Therefore, new product versions can originate several mismatches in the product integrations. Fixing these problems is frequently

postponed due to limited team resources from the companies who are exclusively dedicated to integration projects. The project manager of Company A raised an additional problem: "*my need or urgency may not be the same of my partner*". The companies are aware of these challenges. They currently try to minimise integration problems through follow-up meetings conducted by project managers, who aim at improving the coordination of partners development teams and synchronising developments' prioritisation. However, the companies do not have a clear strategy on how to align their release planning. This situation limits the co-evolution of their products.

D. Requirements Engineering

As market-driven software suppliers, the case companies must understand the needs of market segments to define the product vision and identify market requirements. In Figure 1 we describe the external relationships of the teams from both companies. It evidences the market-driven and customer-specific requirements engineering processes performed by the case companies. It also highlights that several interactions occur between the teams during different phases of the requirements engineering process. This means that partners working in a software ecosystem should make joint decisions regarding the requirements for their products.

At Company A, the product committee and experts in the business domain define the overall product evolution strategy. Moreover, the committee evaluates if requirements defined for a specific integration project can be suitable for a larger group of customers. At Company B, the product manager defines a set of product features in the roadmap to satisfy strategic demands related to current customer market segments as well as targeted market niches. During integration projects, requirements elicitation aims at identifying customisation requests from customers, integration needs from partners, and legal demands. At Company A, requirements elicitation is carried out either by the project manager or business analyst, while the project manager is responsible for this activity at Company B. Company A created an artefact called 'Analysis of Requests and Needs' (ARN), which provides a questionnaire for customers to describe their demands. However, the discourse of the business analyst evidenced flaws in the elicitation process: "*after the system is deployed, the customer sometimes reports that there is something in his process that he did not state; something that was not identified during the elicitation can emerge as a need. This new demand can be analysed and implemented in future releases along the year*".

Companies receive demands from partners to develop new integrations among the products. Integration requirements are jointly defined by team members. These requirements involve technical aspects such as database tables and mappings among systems, data dictionary, data types exchanged and integration flows. Integration requirements must be "*generic and reusable*" among partners, stated a system analyst at Company B. Integration and functional requirements are separated in the project documentation. Interviewees argued that using such simple categorisation accelerates requirements documentation.

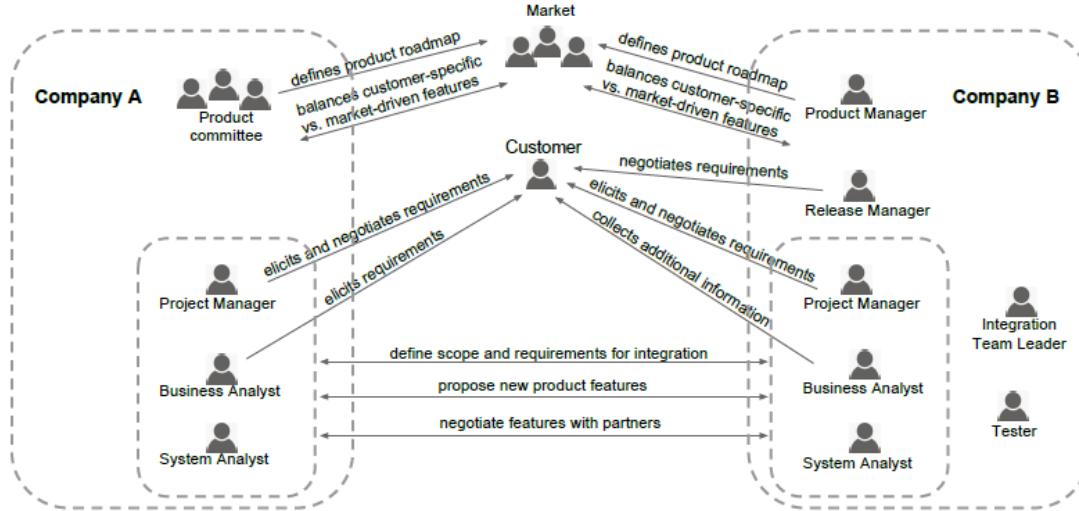


Fig. 1. External Interactions during the Requirements Engineering Process.

A system analyst at Company B explained that integration requirements are identified through functional requirements: *“I need to know the functional requirement to see what is related to the integration. For instance, to create a field in the product registration form I need to define how integrating systems will access it”*. The companies do not formally specify quality requirements, which are simply treated during the development process to address performance and security issues, for instance.

Additional sources of requirements include new ideas provided by project managers, development teams, business analysts, management unities and even the companies' presidents. Once an integration project is concluded, stakeholders from the customer companies can describe their new demands through a web-based request system. The requests are automatically forwarded to the helpdesk service. They conduct an initial filter to select or discard requests. Then, the companies' teams analyse these demands and follow up their status in the tool, from the demand registration until the fulfilment of requirements in a future release.

The case companies face a challenging requirements negotiation due to the interplay with several partners. They must align their own interests and schedules, and negotiate alternative solutions before presenting a proposal to the customer. Negotiation between partners can be either focused on a wider scope of the integration project or it can take place over the set of requirements. According to the project manager of Company A, power conflicts are frequent among partners. Once recognising the business opportunities brought by a specific demand from customers, partners can battle to implement that feature. The requirements negotiation process starts when the project scope is being defined, but it can also happen throughout the integration project. To stay in power, companies frequently attempt to develop features that

they do not have sufficient knowledge to implement. It was highlighted by several interviewees that partners who behave in this manner may harm the success of integration project and damage their reputation.

Agreements are easier to achieve with consolidated alliances, when companies already know their specific roles and duties in the network. In new partnerships, there are uncertainties about the partners' true intentions, yielding conflicts among them. According to a project manager from Company B, *“there is a partnership, but each one has his own goals and interests”*. To treat requirements conflicts, meetings are held to discuss goals, benefits and challenges related to each part of the integration scope. During these meetings, companies negotiate what features each partner will implement. Generally, this decision is based on the companies' technical expertise, but dominant partners can also impose their ambition towards a particular set of features that can be strategic for their product evolution. In addition, each company presents an estimative of time, costs and team resources needed to implement the proposed part of the integration scope. After reaching a consensus, partners individually plan their deliveries. The deliverables are described in the 'Vision Document', which consists of the commercial proposal for a particular integration. Then, the project manager and the release manager negotiate specific requirements with the customer.

Requirements analysis follows the negotiation phase. At Company A, it begins with an initial evaluation by the project manager. She analyses the feasibility of a demand received from the ARN document or the web-based request system. At this moment, no estimation technique is used. Once the demand is considered a suitable requirement, it is registered as a 'Customer Change Request' (CRC) in the One Studio tool. Then, the business analyst evaluates the demand to determine if it consists of evolving an existing feature or creating a new

one. Finally, a planning meeting is carried out among the project manager, business and system analysts. At Company B, the business analyst is responsible for performing such analysis. She verifies whether the requirement is sufficiently described and contacts the stakeholder who demanded it to obtain additional information. Finally, the business analyst verifies whether the requirement shall be introduced as part of the standard product or included in a specific version for the customer who requested it.

Requirements are prioritised in the beginning of each Scrum Sprint at Company A. During a meeting between the project manager and the quality team, requirements are classified as billable, strategic or legal. Requirements prioritisation is carried out by the project manager and the business analyst at Company B. They consider criteria such as customer preferences, impact over the product evolution and dependence with other requirements. Their main concern here is to reduce time-to-market. Considering the joint development, system analysts of partner companies promote meetings to discuss integration aspects such as software artefacts that can be reused and integration requirements. Both companies produce a document describing the technical details of the integration project.

The requirements documentation is validated by the product committee at Company A. After being validated by the project manager at Company B, the documentation is forwarded to the product manager, who analyses the alignment of strategic requirements and macro features with the product roadmap. However, the requirements document is not kept up to date reflecting integrated systems evolution. This hampers the construction of a proper historical basis of integration projects. The project manager from Company A claimed that the main reason for not updating the requirements documentation is due to their very short development cycles. Integration projects bring several small changes in requirements because the evolution of functionalities in one product can affect diverse systems. Hence, the companies develop parameterised functionalities to address the impact of changes on a wide range of users. Changes generated by customer needs and legal issues are generally simple and punctual because of the products maturity.

E. Partnerships

Companies A and B have around 10 and 20 partners, respectively. Generally, partnerships are established due to demands from specific clients to integrate products as well as opportunities to enter in new market segments. However, not all product integrations occur with a partner. In these cases, the supplier simply has to integrate the product with other software solutions developed from other companies or with internal software systems from customers. We identified two types of partnerships between the studied companies: existing products integration and joint new product development. The first case usually happens when a company wins a new contract, then they offer to the customer an integrated solution developed with partners. This strategy enables companies to specialise in a particular domain and indicate partners that complement

their products. It can also happen in a proactive manner, when the companies conduct their annual roadmap planning and identify strategic market niches they want to enter. Then, the companies propose the development of new products with current partners or search for new partners specialised in those segments. The product manager at Company B highlighted that companies usually start the development of new products with partners based on opportunities identified in previous joint projects. Recently, Companies A and B have developed a new product for the pharmacy retail market together with a third partner.

The evolving relationship among partners requires several actions to align business strategies, such as selection of strategic market niches for all parties, negotiation of pricing models, and agreement on how to conduct joint evolution of products and technologies. After establishing a new alliance resulting from an integration project, companies attempt to eliminate overlaps between products and emphasise complementary features. However, as mentioned by the system analyst of Company B, a common challenge is managing the customisation of several products with the specific needs of customers. Partners need to improve the analysis and understanding of the customers' business processes to facilitate the alignment with products features. We identified that integration projects can follow three strategies (Figure 2):

- **Customer-driven Integration** – customers negotiate separate contracts with each supplier. Customers have strong control over the integration project and suppliers have considerable autonomy to evolve their products independently. In this situation, we did not observe an emerging ecosystem being created;
- **Supplier-driven Integration** – customers are aware of the existence of two or more suppliers. However, customers negotiate the contract directly with one supplier. This supplier assumes the responsibility to control the integration project and to conduct negotiations between customers and suppliers. This scenario enables the creation of an ecosystem among actors, in which suppliers have a shared power towards the evolution of software products;
- **Value Added-Reseller** – one supplier assumes the responsibility for the integration with other suppliers by adding features to its own product. The partnership among suppliers is hidden from the customers. The product commercialisation is dealt directly between the customer and the supplier who owns the main product. In this situation, we observed a growing dominance by the central supplier. The creation of an ecosystem depends on the company's ability to demonstrate the benefits of the partnership to other suppliers.

The supplier-driven integration is the most frequent type of partnership in the studied context. Players establish the sharing of duties and start the negotiation to define who is going to implement what features. This is a very critical phase to decide which supplier will control the most strategic features.

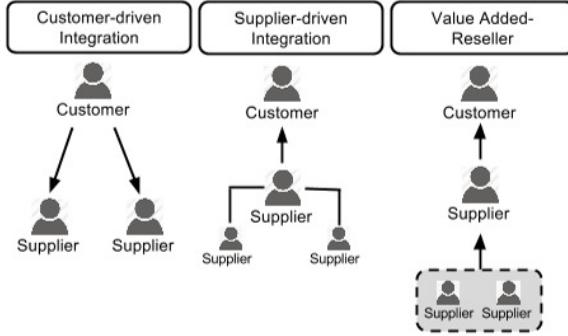


Fig. 2. Partnership strategies.

We observed that technical teams are not explicitly aware of the prosperity of partnerships. Since developers are not involved in strategic decisions, they believe that alliances end after a successful joint integration project, as revealed by one system analyst from Company A: “*this notion of alliance dies once a project is concluded*”. By considering the integration as a temporary effort and not a structural relationship, developers do not understand the long-term opportunities and commitments. In particular, some features may not suit future integrations. Therefore, code quality, evolution, and architectural issues must become structural parts of integration projects with consistent buy-in from the development team.

To address the growing challenges involved in the integration of several products, both companies are developing an integration platform. The companies are negotiating how this platform will be shared and managed by their partners. Top management perceived that the platform is a strategic action to increase the prosperity of their ecosystem. In addition, the release manager and system analyst from Company B are enthusiastic about the benefits of the platform to support the communication among products from different companies. However, other interviewees have different opinions regarding the benefits of the platform. Interviewees from the technical teams of both companies argue that the evolution of the platform is an extra development effort, since the maintenance of the platform is considered quite complex. These divergent viewpoints may indicate the need to better communicate the strategies related to the integration platform. In particular, companies should ensure that decisions about the platform are shared and understood among all levels of the companies.

IV. DISCUSSION

In this section, we present a set of observations to explore the main opportunities and challenges faced by studied companies during the creation of a software ecosystem. These observations are analysed in light of literature in the field.

1. SMEs face a dilemma during the early stages of a software ecosystem: are we competitors or collaborators?

The studied companies confirmed the gains obtained from strengthening relationship with their ecosystem partners. A key benefit obtained from partnerships is the opportunity to

enter in new markets that partners are already well established. In addition, new contracts can be obtained by indication from partners who usually propose joint sales to customers. This means that each partner can specialise in their own competencies to provide complementary features within the ecosystem. The availability of integrated products also means that customers will benefit from time and cost reduction.

However, companies within a software ecosystem face an increasing socio-technical complexity, which not only arises from technological challenges to integrate several software systems but also involves battle for power and control of the most valuable product features. Our findings also indicate that managing joint teams brings a number of challenges. For example, technical teams do not fully understand the purpose, direction and responsibilities for each team; managers responsible for a product integration do not have full authority on the partners teams. Another problem we identified is that technical teams are not fully aware of the relevance of some product integrations. Therefore, studied companies should improve internal and external communication channels to highlight the importance of partnerships.

Participants from both companies recognise the positive aspects of collaboration. Nevertheless, they also acknowledge that competition is inevitable due to the fact that they are medium-sized enterprises fighting to survive in the very competitive Brazilian software market. For instance, product roadmaps and other strategic issues are not fully shared among partners. Companies tend to share more information with long-term partners, while new entrants have very little awareness on the products evolution directions. Similar behaviour has been reported in other studies. In a general business perspective, Moore [16] indicates that the complex interplay of competition and collaboration strategies continues during the lifecycle of an ecosystem. Iansiti and Levien [17] propose that participants in a business ecosystem are engaged in mutually dependent relationships that will affect the health of the ecosystem as well as the individual health of their own business. The level of intimacy wanted in a relationship with companies depends on how critical and strategic the product from a partner is [18]. Therefore, partnerships enable sharing of knowledge, technology and increase innovation potential, factors that makes a company an attractive partner [19]. Coopetition is a main driver of innovation and performance. It takes the relationship between the companies to a new level, where players work together to identify innovative requirements and deliver new solutions that address market needs. According to Levy and colleagues [20], to successfully address coopetition, SMEs need to carefully make decisions on what to share, with whom, when, and under what conditions.

2. Roles are volatile in a software ecosystem formed by SMEs

Manikas and Hansen [9] identified that the most common actors in a software ecosystem are keystone, niche player, external developer, independent software vendor and customer. The keystone plays an active and predominant role in the creation and diffusion of value within the ecosystem. Key-

stone companies often own the platform, which gives them a competitive advantage, but also provide resources to other players contributions in a ‘win-win’ fashion [19].

We did not observe an explicit keystone behaviour in the studied companies. In fact, both companies take the leadership role depending on the market niche in which they are operating. Company B is the most active partner who identifies market opportunities and promotes new alliances. However, Company A has also done so in the past. The shared ownership of the integration platform was considered beneficial by participants to maintain a healthy and balanced partnership. Currently, we did not identify dominators who aim to extract the maximum value of the network without sharing it with other players.

There are several ways to interpret this phenomenon of volatile roles in the ecosystem. The companies are forming a community-oriented ecosystem in which they create flexible committees to promote self-regulation [3]. Other possible explanation is that the ecosystem is not mature enough for a dominator to arise. Another possibility is that an ecosystem composed by SMEs has different dynamics compared to ecosystems created by large organizations who play a central role integrating other players around its own platform. To define a proper direction for the evolution of the ecosystem, the companies need to address several SECO governance issues [21], such as, define clear responsibilities, make business strategy explicit, and decide the level of knowledge sharing. We do believe it is inevitable that the partners will soon observe that their repeated involvement in partnerships requires explicit ecosystem management, as we have analysed in other case studies [22].

3. SMEs participating in a software ecosystem jointly develop and manage a shared platform.

In this scenario of joint development, business and system analysts from both companies participate in several meetings to define integration requirements. These requirements do not refer to feature specification but are rather technical statements to establish the integration process. The impact of integration requirements in all systems must be carefully assessed, since they are part of a shared infrastructure. The teams treat these requirements as reusable assets. By doing that, we can foresee that integration requirements will become platform requirements during the evolution of the SECO. According to Harland and Wust [23], platform requirements involve technical aspects such as software libraries and content databases. The authors claim that platform requirements are the basis of a product platform, which is a development environment used by actors to develop complementary products in the ecosystem.

This discussion of integration requirements is totally aligned with the current efforts of the case companies to build up a central platform that supports their integration projects. Isckia and Lescop [19] describe that a platform enables a composition of functionalities or services that partners can access via a set of common interfaces. The platform will support the development of valuable synergies and complementary innovations for partners and customers. Gawer and Cusumano [24] propose

that the development of a successful platform follows four main stages: define the scope of relationships, build the core strategically, build relationships with external complementors, and optimise internal organizational structures. Iansiti and Levien [17] advocate that a platform is especially important for keystone companies to position their leadership and foster their value proposition in the whole ecosystem. In addition, companies use the platform as an instrument to control their influence on the ecosystem [23].

In the studied companies, an initial platform was created and is being maintained by both partners. They are evolving from a *productisation* to a *platformisation* approach [25], fostering a vibrant and potentially larger ecosystem around the platform. In this sense, the companies are embracing mutual dependency that would require closer alignment of their business models. The companies face a hard decision to make: they should share their internal plans with strategic partners, while they do not want to lose their autonomy. On the other hand, they are aware of the importance of the platform to enable future integrations. Hence, the prosperity of their products will depend on the healthy evolution of the platform. Due to resource constraints faced by SMEs, the attraction of third-party developers can be an important strategy to create niche features.

4. Partner companies must synchronise product strategies to sustain ecosystem success.

Nowadays, software companies are expected to provide an overall view of the product evolution and long-term decision making about future product releases [26]. To effectively integrate products from different partners, companies should make an effort to synchronise their product releases. By doing that, partners can better plan and structure their future releases aligned with the product evolutions from other players. This strategy can also support the maintenance of integrated solutions. Although the studied companies are aware of the problems brought by the lack of product release visibility, they are not fully prepared to evolve their systems in a synchronised and jointly fashion. Frequently, the evolution of one product may generate incompatibilities with other solutions that are part of an integration. For example, there may be conflicts related to features functioning or even removal of features due to potential disuse by another integrated system.

To address these mismatches, partners are gradually aligning features prioritisation in future product releases. In particular, both companies follow agile methods such as SCRUM and Kanban for software development and project management. These practices could also be applied in software product management context. Vlaanderen et al. [27] introduce the notion of product management sprint as a cycle performed immediately before the development sprint. In these product management sprints, partners could not only analyse interdependencies between features, but also start to relate their product strategies. We can foresee that this will lead to the convergence of product roadmaps, supported by the extended release cycles currently targeted by the companies. By synchronising product releases and roadmaps, partners can simplify integrations and establish a self-regulation mechanism. The frequent interaction and

feedback between the companies can promote the openness and transparency of strategies. As a consequence, the health of the ecosystem can be improved.

5. A partner's power has a strong influence on requirements negotiation.

Requirements are negotiated by the case companies in a three-step approach. Primarily, partners define their roles in the integration project and divide the scope of systems integration. Then, the companies identify and prioritise a set of requirements with customers. Finally, a third round of negotiation is performed among partners, who suggest new features for each others products and establish integration requirements. According to interviewees, several challenges arise both during the interactions among partners and with partners and customers.

The tendency of customers to require a full customisation of the products can lead to a tough negotiation. Suppliers must deeply understand the effects of customers' demands to agree on scalable adaptations and reduce maintenance costs, since products are developed for a wider market. However, partners' key challenge consists on disputes for a reasonable division of responsibilities and feature implementation. The strategic positioning of the companies in the software ecosystem strongly affects the negotiation of requirements. This is clearly related to the notion of power, which is a common issue affecting decision-making in requirements engineering [28].

In the studied companies, power relationships are perceived in varied moments. Once inviting a partner to provide an integrated solution and thereby sharing its pool of customers, the company has a greater power over the negotiation. Therefore, the company can select the requirements that aggregate more value for the product. This situation represents the legitimate authority of the inviter company, who may act as a value-added reseller and exert a more pervasive influence over the negotiation process. Another illustrative example occurs when the invitee controls the agenda, since his participation is due to a greater experience on a particular niche or technology. In this case, the direction of power relationships is oriented by the skill or knowledge possessed by the partner, configuring an expert power. Finally, one product may act as a central hub and send information to the other systems. The integration project will be centred around this product, increasing the bargaining power of the supplier during the requirements negotiation. Milne and Maiden [28] call this situation as referent power, which here consists of the strategic characteristics of one product that increase the power of the product's supplier.

We observed that the dynamics of requirements negotiation is constantly changing and this can be attributed to the emergent character of the investigated software ecosystem. According to Moore [16], the lifecycle of an ecosystem involves the phases of birth, expansion, leadership and self-renewal. The case companies are currently building up the software ecosystem by attracting external players to provide solutions that satisfy niche markets. As SMEs, partners aim to complement their products and increase their synergy at this birth stage. However, companies have to face battles for

strategic requirements that allow them to enter in new market niches and thrive in the ecosystem. Such conflicts of interests are very common in emergent ecosystems, since partners have not clearly established power configuration and coordination mechanisms are incipient or even nonexistent. Moreover, the shared responsibility over requirements leads to problems of mutual understanding [29]. To address the positioning issues involved in these negotiations, partners must understand the co-evolution of their products within the ecosystem.

V. CONCLUSION

In this paper, we reported on a case study of two small to medium market-driven organisations that are forming a software ecosystem. The experiences of the studied companies have indicated that by selling their products as complementary solutions to address the needs of specific customers, the companies can access new niches and reduce time to market. Since SMEs face limited resources, tightening collaboration is an important strategy to their survival. However, companies must balance the inevitable competition among them.

The findings from the case study can be summarised as follows. The roles and relationships between partners in young ecosystems are extremely volatile and flexible. The partnership is maintained as long as there exists strategic alignment, translated in efforts to synchronise product releases and roadmaps. Such relationships lead to relatively fast innovations: as collaboration is mostly pragmatic, integrations are built with a short-term view. To counter such downside of the pragmatic view, software vendors can make integration and partnering more structural through SECO governance issues. Accordingly, partners are constructing a shared platform, which shall facilitate the introduction of new features and support the prosperity of integrated products. We also observed that, even in such young relationships, deliberate displays of power and claiming of specific features during requirements negotiation. Companies must share their internal plans and pools of customers with partners, but they do not want to reduce their autonomy. This illustrates a natural part of the process, which is not harmful as long as both companies maintain their benefits. Finally, this paper can be seen as a call to action for software firms to structurally develop their partnering relationships and communication channels, as collaboration seems to be one of the main ways to gain competitive edge in the software sector.

This case study has an explorative purpose. Therefore, we aim to further investigate the presented observations through new studies with SMEs in Brazil and other countries. We plan to explore how other similar software ecosystems deal with the issues investigated in this paper. In addition, with sufficient case material we also aim to create an ecosystem maturity model, with processes and practices that make a company more mature in terms of ecosystem governance.

ACKNOWLEDGMENT

We are grateful to the participants from the studied companies for sharing their experiences and providing valuable

information to this study. This research has been partially funded by CAPES-Brazil (grant 1833-13-8).

REFERENCES

- [1] G. Hanssen and T. Dybå, "Theoretical foundations of software ecosystems," in *Proc. 3rd International Workshop on Software Ecosystems*, 2012, pp. 6–17.
- [2] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," presented at the ICSE Companion, 2009, pp. 187–190.
- [3] G. Hanssen, "A longitudinal case study of an emerging software ecosystem: Implications for practice and theory," *Journal of Systems and Software*, vol. 85, no. 7, pp. 1455–1466, 2012.
- [4] L. Yu, "The market-driven software ecosystem," *IT Professional*, vol. 15, no. 5, pp. 46–50, 2013.
- [5] M. Khalil, P. Dominic, H. Kazemian, and U. Habib, "A study to examine if integration of technology acceptance model's (tam) features help in building a hybrid digital business ecosystem framework for small and medium enterprises (smes)," in *Proc. of Frontiers of Information Technology. FIT'11*, 2011, pp. 161–166.
- [6] J. Bosch, "From software product lines to software ecosystems," in *Proc. 13th International Software Product Line Conference*, 2009, pp. 111–119.
- [7] O. Barbosa, R. Pereira, C. Alves, C. Werner, and S. Jansen, "A systematic mapping study on software ecosystems from a three-dimensional perspective," in *Software Ecosystems - Analyzing and Managing Business Networks in the Software Industry*, S. Jansen, S. Brinkkemper, and M. Cusumano, Eds. Edward Elgar Pub. Ltd., 2013, ch. 3, pp. 59–83.
- [8] F. Santana and C. Werner, "Towards the analysis of software projects dependencies: An exploratory visual study of software ecosystems," in *Proc. 5th International Workshop on Software Ecosystems*, 2013, pp. 7–18.
- [9] K. Manikas and G. Hansen, "Software ecosystems - a systematic literature review," *J. Syst. Softw.*, vol. 86, no. 5, pp. 1294–1306, 2013.
- [10] T. Mens and M. Goeminne, "Analysing the evolution of social aspects of open source software ecosystems," in *Proc. Third International Workshop on Software Ecosystems. CEUR-WS*, 2011, pp. 1–14.
- [11] M. Anvaari and S. Jansen, "Evaluating architectural openness in mobile software platforms," in *Proc. Fourth European Conference on Software Architecture: Companion Volume*, 2010, p. 8592.
- [12] S. Draxler, A. Jung, and G. Stevens, "Managing software portfolios: A comparative study," in *End-User Development*, M. Costabile, Y. Dittrich, G. Fischer, and A. Piccinno, Eds. Springer Berlin Heidelberg, 2011, pp. 337–342.
- [13] P. Runeson and M. Host, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, pp. 131–164, 2009.
- [14] (2013) Weft qda. [Online]. Available: <http://www.pressure.to/qda/>
- [15] (2014) Mps.br. [Online]. Available: <http://www.softex.br/mpsbr/>
- [16] J. Moore, "Predators and prey: a new ecology of competition," *Harvard Business Review*, vol. 3, no. 71, pp. 75–86, 1993.
- [17] M. Iansiti and R. Levien, "Strategy as ecology," *Harvard Business Review*, vol. 3, no. 82, pp. 68–78, 2004.
- [18] J. van Angeren, V. Blieven, and S. Jansen, "Relationship intimacy in software ecosystems: A survey of the dutch software industry," in *Proc. International Conference on Management of Emergent Digital EcoSystems*, 2011, pp. 68–75.
- [19] T. Isckia and D. Lescop, "Open innovation within business ecosystems: A tale from amazon.com," *Communications & Strategies*, vol. 2, no. 74, pp. 37–54, 2009.
- [20] M. Levy, C. Löbbecke, and P. Powell, "Smes co-opetition and knowledge sharing: The is role," in *Proc. 9th European Conference on Information Systems*, 2001, pp. 640–652.
- [21] A. Baars and S. Jansen, "A framework for software ecosystem governance," in *Proc. Third International Conference on Software Business*, 2012, pp. 168–180.
- [22] S. Jansen, S. Brinkkemper, J. Souer, and L. Luinenburg, "Shades of gray: Opening up a software producing organization with the open software enterprise model," *Journal of Systems and Software*, vol. 85, no. 7, pp. 1495–1510, 2012.
- [23] P. E. Harland and S. Wst, "Strategic, brand and platform requirements for an interactive innovation process in business ecosystems," in *Proc. 18th International Conference on Engineering, Technology and Innovation*, 2012, pp. 1–9.
- [24] A. Gawer and M. Cusumano, *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Harvard Business School Press, 2002.
- [25] P. Artz, I. van de Weerd, S. Brinkkemper, and J. Fiegegen, "Production: Transforming from developing customer-specific software to product software," in *Proc. First International Conference on Software Business*, 2010, pp. 90–102.
- [26] T. Suomalainen, O. Saloi, P. Abrahamsson, and J. Simil, "Software product roadmapping in a volatile business environment."
- [27] K. Vlaanderen, S. Jansen, S. Brinkkemper, and E. Jaspers, "The agile requirements refinery: Applying scrum principles to software product management," *Information & Software Technology*, vol. 53, no. 1, pp. 58–70, 2011.
- [28] A. Milne and N. A. M. Maiden, "Power and politics in requirements engineering: Embracing the dark side?" *Requir. Eng.*, vol. 17, no. 2, pp. 83–98, 2012.
- [29] S. Fricker and M. Glinz, "Comparison of requirements hand-off, analysis, and negotiation: Case study," in *Proc. 18th IEEE International Conference on Requirements Engineering*, 2010, p. 167.

The Effect of Variability Modeling on Requirements Satisfaction for the Configuration and Implementation of Off-The-Shelf Software Packages

Amanda Rubython
City&Guilds Kineo
Brighton, UK
amanda.rubython@kineo.com

Neil Maiden
Centre for Human-Computer Interaction Design
City University London, London, UK
N.A.M.Maiden@city.ac.uk

Abstract—An industrial experience of the use of a method for discovering customer requirements with which to configure an off-the-shelf software package for implementation is reported. The method uses an adapted form of product variability model to provide common ground between the customer and supplier about requirements and capabilities. An associated decision support software tool guides the supplier and customer through a model-based walkthrough to discover new requirements, based on equivalent capabilities described in the product variability model. We applied the method in the work processes of the commercial provider of a software-based learning management system, and collected quantitative and qualitative data from supplier-customer interactions. Our first experiences with the method led to an increased exposure and expression of customer requirements in the customer-supplier dialogue, compared to the baseline dialogue during software package demonstrations. The paper also reports some first lessons learned to improve the method and adopt its use with other software supplier organizations.

Index Terms— software packages; requirements-led configurations; product variability modelling; transcript analysis

I. REQUIREMENTS-LED CONFIGURATION OF SOFTWARE PACKAGES

Whilst methods and software tools are now available to support customers to select off-the-shelf software packages that satisfy their requirements [e.g. 1,2,5], there is relatively little research with which to understand how to support software suppliers to adapt (or otherwise) these same off-the-shelf packages, so that a customer's requirements will be satisfied when the package is implemented. Various researchers [e.g. 7] have reported methods and software tools to extract, analyze and manage new requirements on future releases of software packages, however, the role of requirements in the implementation of a selected software package at each customer's site is less understood, and few methods or software tools are available to support it. In this paper, we report experiences with a method based on software product variability modeling and an associated decision support tool that was developed to provide common ground between a software supplier and customer to

articulate and agree requirements in response to possible software package variations.

Although methods and tools now exist, specifying requirements, understanding off-the-shelf software packages and aligning requirements to package features remain difficult tasks for customers to undertake, for at least two reasons. The first is that the information that suppliers often provide about their off-the-shelf software packages is influenced by the need to make sales, rather than to provide comprehensive detail about what the software does and does not do. The second reason, which follows from this first, is that customers often experience problems when determining the extent to which their requirements can be satisfied by a software package's features [1]. One consequence is that more information about the software package needs to be made available, and more guidance about how to use this information needs to be provided. In our work, undertaken on behalf of a supplier of software packages, we sought to develop then evaluate a solution to these problems from the supplier side.

The software supplier, City & Guilds Kineo (CGK), is a global learning and technology company that provides workplace e-learning and learning systems to businesses from industries such as retail and telecoms to travel, electronics and the public sector. The core learning system that CGK offers is called Totara, a configurable, off-the-shelf learning management system. Currently, after a customer selects to purchase or to upgrade the Totara software package, one of CGK's solution architects works closely with customer representatives to understand their requirements so that the package can be configured and/or customized to better satisfy these requirements. The solution architect role combines business analysis and interaction design, and involves liaising with customers to implement Totara into the customer's organization. Such analysis work is increasingly needed to configure and to customize the package's features to satisfy the requirements for the wide range of industries that purchase CGK's products – rarely can one software package satisfy the requirements of such a diverse customer base.

However, CGK's experiences have revealed how little knowledge their customers can sometimes have during the

post-sales process of both Totara's capabilities and their requirements on it. This lack of understanding can lead to unrealistic customer expectations about the system that can cause problems during implementation, and can be attributed to two factors. The first is that some customers fail to have detailed requirements discussions at the purchasing stage due to an over-reliance on Request for Proposals, with the decision to buy based on how CGK responded to a list of customer requirements in writing, rather than a practical assessment of Totara capabilities per se. The second factor is that the dialogue between CGK and the customer at the sales stage is usually supported by software demonstrations that do not illustrate specific system features and configurations to meet customer needs. Moreover, the inevitable time and resource pressures on both supplier and customer imposed by a procurement exercise can exacerbate both of these factors.

Therefore, to improve customer understanding of Totara and to support requirements specification during the CGK sales process, we developed a new method to provide a customer with explicit information about requirements-level capabilities that Totara was originally developed to provide to users. The method is new in that it combines variability modeling from software product line engineering with text-based requirements and use case specification techniques to describe possible variations in the configuration of Totara in a form that non-technical customers can understand. We sought to explore, through studies conducted as part of CGK's sales activities with its own customers, whether the method could:

- Q1: Expose more customer requirements on the Totara software package implementation, compared to the current software package demonstration process?
- Q2: Improve customer understanding of software package capabilities, compared to the Totara current software package demonstration process?
- Q3: Allow supplier and customer to align customer requirements and software package capabilities more effectively, compared to the Totara current software package demonstration process?

The remainder of this experience report is in 4 sections. Section 2 outlines the method, called *ETHER*, introduced into CGK's sales work process, and selected variability modeling developed for the Totara software package used in the method. Section 3 reports experiences from the use of the method with 4 different CGK's customers, as well as a comparative baseline process undertaken with a 5th customer without the *ETHER* method. Analyses of the transcripts of the customer representative-solution architect discussions are reported to seek first answers to the above questions. Section 4 reports lessons learned from the analyses of the experiences, and outlines both future research opportunities and the next steps for CGK.

II. THE *ETHER* APPROACH

Currently, when selecting the Totara software package for their organization, customers ask CGK for information about the package. A sales consultant is responsible for investigating requirements with the customer, providing information about Totara to the customer, and costing the proposal. However, when a customer requests more information about one or more Totara features, a solution architect is called upon to demon-

strate features to customer representatives. This normally involves an optional short exploration of the customer requirements, then a detailed presentation of the feature and its functions in the software package that lasts about one hour.

We developed the *ETHER* (*Expose THE Requirements*) method to replace these software-led feature demonstrations with more effective, requirements-level communication about how the system can be configured to meet customer requirements more effectively, as well as to allow CGK to produce more accurate requirements-based costings for customers. The *ETHER* method can be applied either in a face-to-face meeting or in a webinar in which the solution architect and customer representatives explore requirements, package capabilities and their alignment. The method is composed of a four-step process and a decision support software tool that was developed to support some of these steps. Each of the four steps of the method is outlined in turn.

A. Step 1: Introduction

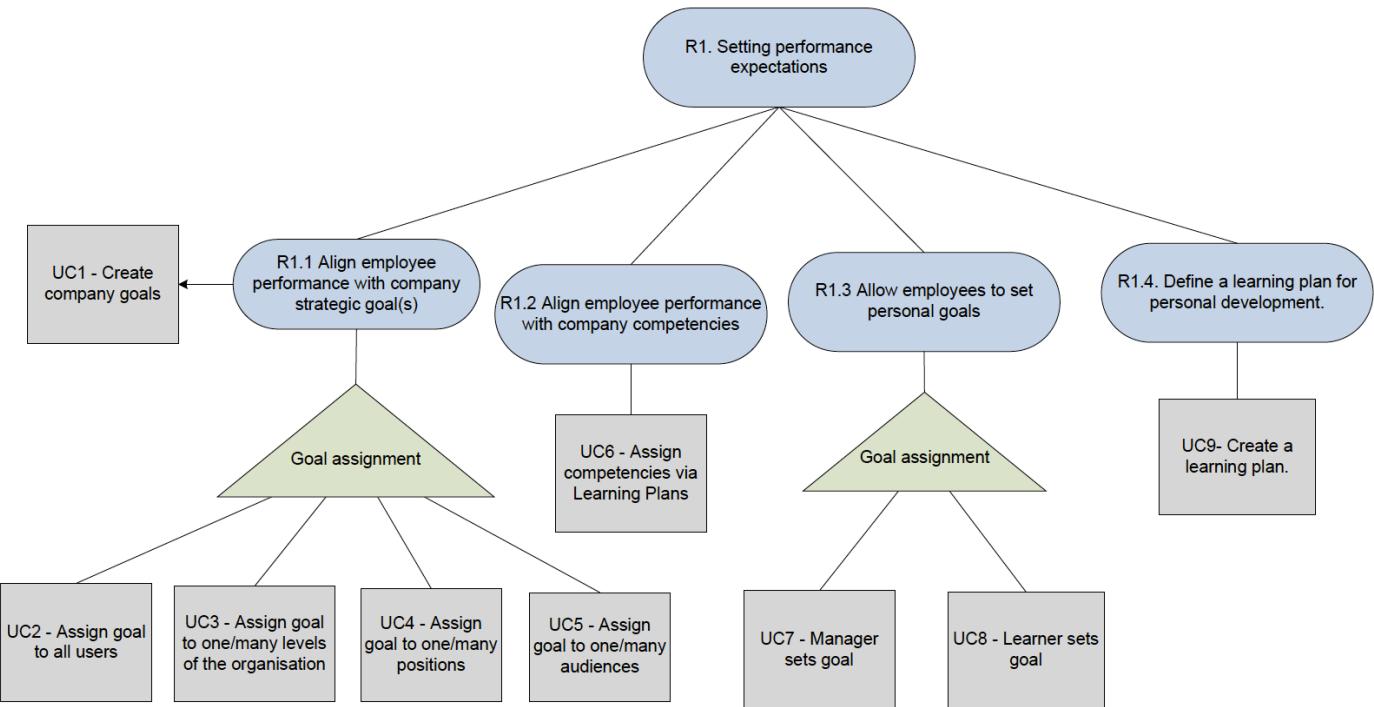
First of all, the solution architect presents the Totara features to be explored, and sets customer expectations – not all customer requirements can be satisfied by a feature, so more expensive software customization and/or business process change might sometimes be needed.

B. Step 2: Build Common Ground

Requirements information is exchanged between customer representatives and the solution architect to build common ground needed for later communication. Common ground is defined as what two or more individuals jointly know or believe [4, p93] – a form of common self-awareness – and communication between people cannot be effective unless all participants share the same perception of knowledge and beliefs [4, p120]. Therefore, the solution architect seeks to build common ground about the customer's requirements and capabilities of Totara to satisfy these requirements, independent of the design and implementation of the software features. To do this, the solution architect first elicits relevant business requirements from the customer representatives, then shares a glossary of relevant Totara terms to avoid potential miscommunication about the software package. Afterwards, the solution architect verbally provides the customer representatives with a high-level description of package capabilities, to prepare the customer for the more in-depth analysis in the next step.

C. Step 3: Requirements that the Product Supports

Then, rather than walk through relevant software features without reference to requirements as undertaken in the current process, the solution architect uses *ETHER* to walk the customer through a bespoke support tool with which to guide decision-making about requirements on the Totara implementation. The decision support tool guides the solution architect and customer representatives through key requirements-based decisions to make about selected Totara features. Support for each decision to make with the tool is based on an underlying goal variability model that was developed as part of the method. This goal variability modeling is based on variability models from software product lines engineering, a paradigm which aims to reduce time and cost in software development by reusing components across a product line [9]. Each variability



model shows where variations can occur in the product line and why, as well as the relationships of dependency and constraint [3]. In *ETHER*, each goal variability model specifies the possible variations in the capabilities of Totara features, independent of their software implementation. The solution architect uses the model and decision support tool to guide the decision-making about requirements that can be satisfied by Totara.

For example, to create a goal variability model for a Totara feature such as *performance management*, a CGK solution architect first defines the requirements achievable using the feature, then maps these requirements to one or more components of the feature, and to a sub-requirement that was a means to achieve the requirements. For each sub-requirement that is not further refined into requirements, a use case was specified of the step-by-step actions using Totara to achieve the requirement. The resulting model for one top-level requirement related to the *performance management* feature of Totara is depicted in Figure 1. The model specifies four variations to achieve the requirement *R1. Setting performance expectations*. One variation is *R1.1 Align employee performance with company strategic goal(s)*, which is in turn achieved by completing the use case *UC1 Create Goals* and assigning goal, which in turn be accomplished in 4 ways represented by 4 use cases.

Whilst it is possible that each goal variability model, on its own, could provide common ground in discussions between the solution architect and customer representatives, first experiences with such models indicated that its form was too difficult for most customer representatives to understand. Therefore, the *ETHER* method was extended with the decision support software tool that presents each variation point in each model as a requirement-based decision to make to the solution

architect and customer representatives. During this step, the solution architect and customer representatives walk through decisions about product capabilities, guided by *ETHER*'s decision support tool. For each decision point, which is equivalent to an alternative capability in the model achieved by use of the Totara package, the solution architect describes alternative capabilities, and asks which of these align to one or more customer requirements. If the customer's answer is yes for any of the capabilities, then the solution architect selects that option in the decision support tool, and explores each set of associated lower-level capabilities using the same process. At the lowest-level node of each hierarchy of capabilities is a description of a use case that specifies the steps needed to perform a task to deliver the capability – steps that the architect could later demonstrate with the software package, beyond the current scope of the method. Examples of alternative capabilities of Totara's *performance management* feature presented in *ETHER*'s decision support tool are shown in Figure 2. The left-hand side shows alternative capabilities, and the right-hand side alternative means to achieve one capability.

D. Step 4: Session Review

After all alternative requirements-level capabilities for the feature have been walked through, the solution architect asks the customer representatives if there are any remaining requirements to investigate. Once these other requirements have been considered, the solution architect produces a written summary of the session and sends it to the customer for agreement.

R1: Setting Performance Expectations
How performance standards are defined and assigned to employees.

Do you want to:

R1.1 Align employee performance with company strategic goal(s)
Does your company have a set of company goals which are assigned to employees as part of the appraisal process?

R1.2 Align employee performance with company competencies
Does your company have a set of competencies which describe a level of skill or behaviour which should be attained and are assigned to employees as part of the appraisal process?

R1.3 Allow employees to set personal goals
Are employees set individual goals?

R1.4. Define a learning plan for personal development
Does your company set up a plan of training or development opportunities as part of managing performance?

Requirements Overview

Yes - show me more options

R1.1: Align employee performance with company strategic goal(s)
A goal framework needs to be created before company goals can be assigned.

[Create company goals - \(UC1\)](#)

Once created goals can be assigned in one or many of the following ways:

Assign a company goal to all users - UC2
Requires an organisation or position framework or audiences.

Assign a company goal to one/many levels of the organisation - UC3
Requires an organisation framework.

Assign a company goal to one/many positions - UC4
Requires a position framework.

Assign goal to one/many audiences - UC5
Requires audiences.

Look at another requirement

Figure 2: Two screenshots from the *ETHER* decision support tool that replaces demonstration of concrete software features. The left-hand side shows alternative capabilities to achieve *R1: Setting performance expectations*. The right-hand side depicts the lower-level capability *R1.1 Align employee performance with company strategic goal(s)* has been selected

E. Delivering a First Version of the *ETHER* Method

We developed a first complete version of the *ETHER* method, with descriptions of the steps and a prototype decision support software tool, and made it available to CGK for use by its solution architects. Experiences of use of the method are reported at length in the next section.

III. FIRST EXPERIENCES WITH THE *ETHER* METHOD

In order to evaluate the *ETHER* method, the first author organised sessions with 5 CGK customers to present the *performance management* feature available in Totara version 2.5. At the time of the evaluation, Totara 2.5 was scheduled for release to customers, so each session offered an early opportunity for each customer to preview the new feature and to start to make budgetary decisions regarding whether to upgrade or not their version of Totara. The 5 customers were recruited via the CGK sales team as customers who had previously shown interest in implementing appraisals and performance reviews into their learning management systems. They had very little knowledge of what capabilities the new *performance management* feature consisted of. Each session was run as either a face-to-face meeting or a webinar depending on the customer's preference. One solution architect facilitated each session. The first webinar session was undertaken as a baseline session in order to capture data about the current demonstration process used in CGK that data from the other sessions would be compared to. During the other 4 sessions, the solution architect applied the *ETHER* method. Three of these sessions were webinars, and 1 was conducted in a face-to-face meeting. Three different CGK solution architects undertook the 5 sessions with 5 different customers in 5 different sectors, and the durations of the sessions was from 45 to 143 minutes, see Table 1.

Each of the 5 sessions was successful, in that the solution architect was able to demonstrate software features of Totara in session 1, and step through all 4 steps of the *ETHER* method in the other 4 sessions to investigate alternative requirement-level

capabilities. The one failure to apply the *ETHER* method as specified occurred in the first step of all 4 sessions, when the solution architects did not support their verbal summaries of each session with the planned written reports due to lack of resources to complete the tasks.

Interviews undertaken by the first author with solution architects after each session revealed some first perceived advantages and disadvantages of the *ETHER* method over the current software package demonstration process. After session 2, for example, the solution architect reported that: “*Yeah it did help more than doing a demo without it in the sense of I feel that after that amount of time I have a better idea of what they are trying to achieve then when I don't use something that is this structured and then do a Totara demo*”. Some of the customer representatives also reported benefits, for example: “*.. opportunity to see the system at a high level in order to think about/initiate discussion around our own business requirements*”. That said, after the third session, the solution architect stated his belief that some of the method steps were more useful than others: “*I find the walkthrough really, really useful for the first half but for the second half I feel that you don't have to use it as much, more as a prompt so when I'm like ah okay which bit do I need to focus on next I go back to it....*” The use of the glossary of terms to describe the software package was important during step 2 for building common ground: “*I think internally as they didn't have terminology for some of these things it was more an agreement of OK this is what it means. I'm not certain it made a great deal of difference when I was doing the demo. It is a useful thing to do and also introduces them to the terminology. It introduces them to the characters they will come across in the demo... you know here are a list of people we are dealing with – that sort of thing*”. Establishing the glossary of package terms also appeared to improve understanding: “*Yes so, well I ran through the glossary, terminology thingy, they call them what we call them, but that was actually quite useful as they said we don't*

use that, we don't do that so that set my expectations for what I then covered, so what do you call competencies, we don't call them anything, we don't use them".

Table 1: Comparative data about the 5 solution architect-customer representative sessions

| Session | Led by | Number of customer attendees | Customer industry | Format | Duration (mins) |
|-----------|--------|------------------------------|----------------------------------|--------------|-----------------|
| Baseline | SA3 | 3 | Training Provider | Webinar | 60 |
| Session 1 | SA1 | 2 | Catering / Facilities management | Webinar | 45 |
| Session 2 | SA2 | 1 | Newspaper /Magazine wholesaler | Webinar | 143 |
| Session 3 | SA1 | 3 | Property Development | Face to face | 73 |
| Session 4 | SA2 | 2 | Retail | Webinar | 75 |

That said, the solution architect in the first session reported that it: *“.. took longer to do the demo than originally thought as you need to cover areas that they don't know about e.g. learning plans (which leads onto questions about programs) and competencies. These in themselves open further questions so it's easy to be drawn into other areas of Totara..... In general it covered everything they needed to know, but it was quite high level. Will definitely require follow up demos more targeted at their actual requirements”*. The comment indicated that the solution architects might have preferred to combine use of the *ETHER* method and tool with the demonstration of software package features.

The use of the decision support tool for the method was perceived to aid the process, but the potential for further features was identified. One solution architect reported that: *“using the tool gives a nice structure to the demo although some parts raise questions about other areas that I covered later, so you need be able to answer questions early, or manage the demo to say it's coming later. Might be worth adding in additional links to jump around the structure to get to related areas so that the demo isn't so prescriptive in its flow and order”*. Another solution architect identified the tool as useful in parts of the session: *“so actually probably a good high level comment that I have is this feature – this – this section – performance expectations was really good and really helped, but facilitating the appraisal process I didn't really use it for, the reason why was because this one is more about the consultancy of how we are going to do it, whereas facilitating the appraisal process was more you set up the form, you set up the stages, there is not so much consultancy.”*

Overall, the feedback from these interviews revealed both the potential benefits and the limitations of the first version of the *ETHER* method. However, to investigate the effectiveness of the method in more detail, we audio-recorded the solution architect-customer representative dialogue in each session. This dialogue was the primary means of exposing requirements then communicating and aligning them with product capabilities in face-to-face meetings and webinars, therefore a first-hand analysis of the dialogue was expected to reveal new

insights into use of the *ETHER* method. Each full audio recording was transcribed, and information from the screen captures from the decision support tool was described and added to the transcript. We then applied a simple thematic coding to each transcript segment. The set of codes generated from the research questions and reported in Table 2 was developed and applied to each transcript from each session.

Table 2: Thematic codes used to analyze the transcripts of the 5 solution architect-customer representative sessions

| Code | Description |
|--------------------------------------|--|
| Requirement | An expression of capability and/or quality needed or wanted by the customer |
| Requirement-capability alignment | An expression of a requirement's alignment to one or more product capabilities |
| Requirement-capability non-alignment | An expression of a requirement's explicit non-alignment to one or more product capabilities |
| Customer understanding | An explicit expression of understanding of one or more package capabilities, features or qualities |

To analyze the codified transcripts, the totals of instances of each type of coded segment were computed and compared across the 5 sessions. We then undertook a content analysis for each codified transcript segment to compute the totals of requirements. In the next sections, we report key results from this analysis of the data collected from the 5 sessions.

A. Exposed Requirements

First of all, we applied results from the contents analysis to calculate the totals of different atomic requirements that were articulated in each session. We also calculated the totals of these articulated requirements that were stated to be satisfied by the package capabilities, and the totals stated explicitly to be unsatisfied by these capabilities.

In the baseline session, the customer representatives and solution architect did not articulate any requirements in the one-hour session, a result consistent with previous CGK experiences. In contrast, in the 4 sessions during which the *ETHER* method was used, the customer representatives and solution architects articulated a total of 67 different atomic requirements – an average of almost 17 requirements per session, see Table 3.

Table 3: The totals of different atomic requirements articulated in each of the sessions, and the totals of these requirements explicitly matched to package capabilities, and explicitly not matched to package capabilities

| Session | Totals of Articulated Atomic Requirements | Totals of Requirement Capability Alignment | Totals of Requirements Capability Non-Alignment |
|-----------|---|--|---|
| Baseline | 0 | 0 | 0 |
| Session 1 | 18 | 3 | 0 |
| Session 2 | 29 | 8 | 4 |
| Session 3 | 16 | 3 | 0 |
| Session 4 | 4 | 2 | 1 |

That said, the sessions varied – session 2 led to the articulation of 29 requirements, whereas session 4 led to the articulation of only 4 requirements. Both of these sessions were ran by the same solution architect, but session 2 ran for twice as long as session 4. Of the total of 67 requirements, only 16 (24%) were aligned to package capabilities in the sessions, 5 (7%)

were recognised as unaligned to any package capabilities in the sessions, and the remaining 44 (69%) were not associated to any package capability. This result revealed that the sessions with the *ETHER* method did not manage to associate most customer requirements to the package capabilities.

In contrast, the *ETHER* method did appear to increase the number of customer requirements articulated, compared with its current software package demonstration process, even if most customer requirements were not explicitly identified as satisfied or not by the package capabilities. To explore possible reasons for this, we examined which steps of the *ETHER* method each requirement was articulated in.

B. Where and How Requirements were Exposed

The totals of the requirements that were articulated in the 4 steps of the *ETHER* method are reported in Table 4. The majority – 46 (69%) – were articulated in step 2 – building common ground – when the solution architects asked customer representatives to express their requirements. The remaining requirements were articulated first in step 3 – requirements that the product supports – revealing that the role of the decision support tool to explore alternative package capabilities and surface new requirements. No new requirements were articulated during introductions and reviews in steps 1 and 4.

We also undertook a simple content analysis of all of the 67 requirements that were articulated during the 4 sessions, to understand the nature of the requirements exposed by use of the *ETHER* method. To do this, we categorized each of the 67 requirements as a member of one of 3 possible types:

- *An independent requirement*: a customer representative states explicitly that the customer wants a capability, function, feature or quality, independently of any package capabilities, for example *we need to do x, if it could do x, and they would need x*;
- *A deficient business process*: a customer representative expresses what is needed in terms of deficiencies in the current process. Examples include: *so at the moment we do x*, to describe the process, and *we have x*, to describe an element of the process, such as a form or an element of the current system in place;
- *A requirement that is generated directly from package capabilities*: these types of requirements arise from the dialogue about package capabilities. The customer representative makes a statement about the capability and how they will use it, for example: *ok so I can have x*, to describe what the customer would like, and *so if the package does x, then we can do y*.

Table 4: Totals of different atomic requirements reported in each of the 4 steps of the *ETHER* method, in each session

| Session | Step 1 | Step 2 | Step 3 | Step 4 |
|--------------|----------|-----------|-----------|----------|
| Session 1 | 0 | 12 | 6 | 0 |
| Session 2 | 0 | 19 | 10 | 0 |
| Session 3 | 0 | 12 | 4 | 0 |
| Session 4 | 0 | 3 | 1 | 0 |
| Total | 0 | 46 | 21 | 0 |

Table 5 reports the total number of occurrences of each type of requirement in the 4 sessions.

Table 5: The totals of different atomic requirements, by type, reported in each session

| Session | Independent requirement | Deficient business process | Requirement generated from package capabilities |
|--------------|-------------------------|----------------------------|---|
| Session 1 | 2 | 14 | 2 |
| Session 2 | 6 | 19 | 4 |
| Session 3 | 0 | 14 | 2 |
| Session 4 | 1 | 3 | 0 |
| Total | 9 | 50 | 8 |

Most of the requirements, 50 of the 67, were expressed as deficiencies in current work processes to resolve, as this extract from session 2 demonstrates: *“At the moment we use a separate system...and basically we have an online scorecard where people can create online goals - the executees create their goals at the start of the financial year which is the 1st September so we’re just beginning that process now. Those goals are cascaded to senior team and then they are cascaded down the organization and then individuals are asked to go in and amend the goals to make them more appropriate and applicable to themselves basically”*. In this event, 4 requirements were exposed – for executees to create performance objectives for the company and assign objectives to the senior management team, for the senior management team to assign objectives to individual employees, and for individual to amend objectives.

In contrast, the customer representatives expressed many fewer requirements independent of the package capabilities, for example: *“We need to be mindful that that human element is going to be taken away we need some robust logic that says this equals that and therefore you get this document rather than their manager saying I’m not really sure”*, and *“I’m wondering whether we could set a learner or a user up to create their own 360 forms so perhaps their manager approves it or something. That might be quite useful.”* These new requirements specified the role for learners to create their own 360 forms and for managers to approve these forms.

More surprisingly, only 8 of the 69 requirements were generated directly from package capabilities. For example in session 2, the customer recognizes a requirement when the options for assigning a company goal are shown on screen: *“I think we’d want to do both really depending on who the learner is sometimes it would be learner driven sometimes it would be manager driven. The ideal is that all learners go out and manage objectives but I didn’t that’s reality so I think flexibility to do both please.”* The customer generated this requirement from capabilities of the package, in contrast to software demonstrations in which the customer was not encouraged to compare capabilities directly to their business activities. However, such instances of requirements generation were few and far between during the sessions.

A qualitative analysis of the relevant transcript segments revealed that each statement of satisfaction between a requirement and a package capability emerged in one of two different ways. In the first way, the solution architect or one of the customer representatives would refer back to a requirement articulated in step 2 when common ground was established, then aligned or otherwise that requirement to one or more capabilities of the package, for example in session 1 the solution archi-

tect says: “*So the defining the appraisal is what we just run through, so it setting up those stages, setting up those pages to view, assigning appraisals as I just mentioned you can assign using audiences positions and organization hierarchies so we can get the right things to the right people. As you say you've got multiple forms so what you can do within the system is create multiple appraisals*”. Often the solution architects appeared to recall the customer requirements because they did not make external notes of customer requirements during the sessions.

The second way in which a statement of satisfaction was made was to align a new requirement directly to one or more package capabilities immediately in the session. Both the solution architects and customer representatives articulated compliance statements when recognizing that a new requirement that might align to a capabilities currently being described. For example, one of the customer representatives asked: “*is there a way of completing appraisals off-line at some point? For example they use down time whenever they can they may not always be on the Internet, they may be on a train or somewhere else and maybe not necessarily now, but in the future will there be some sort of functionality to do stuff off-line because some people have teams of about 50 people and it's such a huge task for them*”. The solution architect responded: “*Yeah yeah I know, I can definitely see why. Unfortunately no, as you have said the site is web-based so it does require the Internet...*”, thereby articulating an explicit lack of alignment of that package capability to the requirement.

C. Customer Understanding

The coding of the transcript segments to reveal evidence of understanding package capabilities by customer representatives summarized in Table 6 revealed that the customers articulated all but one statement of understanding in step 3 exploring requirements supported by the package. In contrast, the customers did not make such statements during the other 2 steps of the method.

Table 6 - Where understanding occurred

| Session | Step 2 | Step 3 | Steps 1 and 4 |
|--------------|----------|-----------|---------------|
| Session 1 | 1 | 8 | 0 |
| Session 2 | 0 | 1 | 0 |
| Session 3 | 0 | 4 | 0 |
| Session 4 | 0 | 0 | 0 |
| Total | 1 | 13 | 0 |

Moreover, most of the statements of understanding were expressed as questions such as: “*hmmm, so we can have lots of different messages going out at different stages...?*” The statements were often prompted by information displayed on the screen shared by the solution architect and customer. For example, in session 1, the solution architect explained: “*...but what I can do is add a specific, those are the company goals, I can add in a goal from a manager*”, to which one of the customer representatives replied: “*so those goals are added based on the audience those goals area added based on the audience automatically?*”. The solution architect then responded: “*yup, I think actually, this is where I may get a little bit vague. I think what it might be is when you... I think on this version when you go in, you have to manually add them in. My understanding is for the release version it will automatically pull them in for you*”.

To conclude, analysis of the dialogue transcripts revealed little direct evidence of customer understanding, although there was also little direct evidence of misunderstanding between each solution architect and customer representatives.

IV. REVISITING THE QUESTIONS, AND LESSONS LEARNED

We reviewed the outcomes from this first use of the *ETHER* method against the 3 questions, presented at the start of the paper, to extract some first lessons learned for both CGK and for requirements research applied to implementing off-the-shelf software packages.

To answer the research question Q1, the results indicated that use of the *ETHER* method did expose more customer requirements on the implementation of the software package compared to CGK’s current demonstration process. Both CGK’s past experiences and the baseline session revealed that few requirements are exposed and documented during demonstration sessions. In contrast, the 4 *ETHER* sessions exposed a total of 67 customer requirements. In simple terms, the sessions revealed that replacing the software package implementation with decision support about requirements-level capabilities increased the exposure of requirements in the dialogue between supplier and customer. Although perhaps not the most surprising result, the result does indicate that the level of abstraction at which information is presented to customers has the potential to influence the discovery of and communication about requirements. However, most of the exposed requirements were expressed not as direct features or qualities that the implementation should have, but in terms of deficiencies in each customer’s current business process that could be enhanced by the Totara package. The customer representatives expressed most of these requirements in a relaxed and natural way, perhaps in part because they expected the CGK solution architects to formalize the requirements afterwards on their behalf. Indeed, one interpretation of this behaviour was that the customer representatives were waiting to understand the package capabilities before seeking to formalize requirements further. For example, the customer who participated in Session 2 requested a follow-up requirements session before finally deciding to buy the *performance management* feature. One broader implication is that the *ETHER* method needs further guidance to extract requirements from deficiencies in the customer’s current situation. We explore this implication further in the lessons learned reported at the end of this section.

To answer the research question Q2, the use of *ETHER* revealed only limited evidence of improved understanding of software package capabilities by the customers, compared to CGK’s current software package demonstration process. One possible reason for this result is the good level of understanding that the Totara customers already had of these capabilities – customers in all 5 sessions reported that the sessions were ‘*very effective*’ for ensuring understanding of the *performance management* feature.

Finally, to answer the research question Q3, the use of the *ETHER* method did not appear to allow the supplier and customer representatives to align customer requirements and software package capabilities more effectively, compared to the current software package demonstration process. Whilst there was an increase in the number of requirements that were both explicitly aligned to and not aligned to package capabili-

ties in *ETHER* sessions over the baseline session, this increase was not large, and most requirements that were exposed during the sessions with the *ETHER* method were not either explicitly aligned or unaligned with the package capabilities. The original design of the *ETHER* method assumed that customers would provide well defined and clearly stated business requirements at the start of each session. The result indicates that this did not happen. Instead, the method will need to provide more support for aligning requirements and capabilities.

From the results, we drew a number of simple lessons learned that can be applied first by both CGK and researchers seeking to align off-the-shelf software packages and requirements from customers more effectively. The first lesson is that supporting software package demonstrations with equivalent artifacts that describe the requirements-level capabilities of these packages can lead to the exposure of more customer requirements on packages. In this work we report the combined use of an adapted product variability model and explicit software decision support tool to walk customers through the capabilities of a system and the requirements that it supports. Of course, other types of artifact that depict goals and requirements which are reverse-engineered from software package features can also be used, but the focus of goal variability models on decision points made them an effective mechanism with which to focus on requirements-based decisions to make.

The second lesson is one that will be applied to future uses of *ETHER* in CGK, who will extend the method with one of more mechanisms with which to structure and record customer requirements more explicitly. This explicit record of each requirement will be needed, we believe, to facilitate the more effective alignment of requirements to the software package capabilities – one area of method use that was weak in the reported experiences. Existing requirements templates such as VOLERE [8] and writing guidelines such as EARS [6] can be used to guide customers to write and structure key requirements before each session and during step 2 when building common ground, although care will be needed to ensure that this change does not result in a long list of detailed requirements from the customer and a transactional approach similar to the Request for Proposal process. Moreover, the method will need to provide guidance to extract clear requirements statements from descriptions of business process deficiencies. The session dialogue between the solution architect and customer representatives can then extended with an explicit handshake process, in which the solution architect verbally presents new customer requirements back to the customer representatives during the session. Simple tabular or graphical representations can be used to externalize and encourage alignments to be

specified between requirements and package capabilities. These representations can be supported digitally to allow solution architects to align emerging requirements with modeled package features.

Finally, a wider possible change is to extend the scope of the *ETHER* method beyond the sessions involving the solution architect and customer representatives. For example, rather than present the software package capabilities to customers during each session with the decision support tool, the goal variability model and/or decision support tool could be provided to customer representatives to familiarize themselves with the key decisions to make, prior to the session. Of course, presenting different package capability options to customers without expert guidance from suppliers would create risks, and necessitate additional documentation about the software package to be delivered.

We are looking forward to implementing these changes, and reporting on their use in the near future.

ACKNOWLEDGMENT

We acknowledge the support of both City&Guilds Kineo and their customers to undertake the evaluation studies reported in this paper.

REFERENCES

- [1] Alves, C., Finkelstein, A., 2002. 'Challenges in COTS decision-making: a goal-driven requirements engineering perspective', in: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering. pp. 789–794.
- [2] Alves, C., Franch, X., Carvalho, J.P., Finkelstein, A., 2005. 'Using goals and quality models to support the matching analysis during cots selection', in: COTS-Based Software Systems. Springer, pp. 146–156.
- [3] Bühne, S., Lauenroth, K., Pohl, K., 2004. 'Why is it not sufficient to model requirements variability with feature models', in: Proceedings of the Workshop: Automotive Requirements Engineering (AURE'04), Co-located at RE'04, Nagoya, Japan.
- [4] Clark, H., 1996. 'Using language', Cambridge University Press.
- [5] Maiden N.A.M. & Neube C., 1998, 'Acquiring Requirements for Commercial Off-The-Shelf Package Selection', IEEE Software, 15(2), 46-56.
- [6] Mavin A., 2009, 'Easy Approach to Requirements Syntax (EARS)', Proceedings 17th IEEE Requirements Engineering Conference, IEEE Computer Society Press, 317-322.
- [7] Natt och Dag J., Gervasi V., Brinkkemper S. & Regnell R., 2005, 'A Linguistic-Engineering Approach to Large-Scale Requirements Management', IEEE Software 22(1), 32-39.
- [8] Robertson, S., Robertson, J., 2013. 'Mastering the requirements process: getting requirements right', 3rd ed. ed. Addison-Wesley.
- [9] Schmid, K., Santana de Almeida, E., 2013. 'Product Line Engineering'. Softw. IEEE 30, 24–30.

Modelling Sustainability in a Procurement System: An Experience Report

Camilla Bomfim¹, Wesley Nunes¹, Leticia Duboc¹

¹Dept. of Computer Science

State University of Rio de Janeiro, Brazil

camillajbomfim@gmail.com,

aleysenun@gmail.com, leticia@ime.uerj.br

Marcelo Schots^{1,2}

² Systems Engineering and Computer Science - COPPE/UFRJ

Federal University of Rio de Janeiro, Brazil

schots@cos.ufrj.br

Abstract—Sustainability is one of the main driving forces in our society. IT can also contribute to sustainable development, which goes beyond the energy consumed to produce and run the software product. The development of social-technical systems can have a significant impact on the sustainability of its surroundings. A particular type of system with considerable sustainability impact are procurement systems. They normally affect the three pillars of sustainability: social, economic and environmental. This paper describes an experience on using goal modelling to incorporate sustainability into the procurement system of a large multinational energy company. The study highlights the advantages and challenges of introducing sustainability into private procurement systems, as well as the suitability of the technique for such a purpose. We believe this experience and its resulting model can be useful to other companies wishing to implement sustainable procurement processes.

I. INTRODUCTION

Sustainable development is defined as “the ability to meet the needs of the present without compromising the ability of future generations to satisfy their own needs” [1]. Sustainability has become a driving force in our society. Concerns about the impact of human activities have been increasing, accompanied by worldwide efforts to achieve sustainable development [2]. These concerns have started to reflect in organizations. Some find themselves obliged to comply with regulations in order to operate, such as organizations that exploit natural resources [3]. Others have begun to understand the importance of having a public image for “sustainable organization”, as consumers are increasingly preferring products and services that have a sustainable character [4].

A survey with around 1,000 CEOs in 2010 [5] showed that their majority (93%) reported sustainability to be “important” or “very important” to the success of their business and that it should be fully incorporated into strategies and operations (96%), implemented by boards (93%), and integrated in the supply chain (88%). In 2013 [6], the survey showed similar results: again 93% of the CEOs regarded sustainability as key to success. Nevertheless, a deeper analysis of the data in this latest survey reveals that business efforts on sustainability may have reached a plateau instead of continuing for a new peak, perhaps reflecting the current economic climate. In spite of the doubt about the pace of change and the scale of impact, the survey also reveals that “the corporate sustainability movement

is broadening, with a deeper awareness and commitment evident in every quarter of the world” [6].

IT has an important role to fulfil in sustainable development. First, IT systems are large consumers of resources, not only when they are operating, but also considering their whole life cycle [7][2]. Second, IT is at the heart of many areas critical to sustainability and can be crucial in creating awareness for sustainable practices, as well as in facilitating the implementation and monitoring of strategies for sustainability.

A particular type of IT system that considerably impacts on the three pillars of sustainability (social, economic and environmental) are procurement systems. Such systems can be used by both private and public organizations to support the processes for purchasing products and contracting services. Usually, these systems are not big energy consumers, yet they can have a considerable influence on the sustainable development of the communities around them.

Governments are normally interested in promoting sustainable development, which led to the creation of international guidelines for sustainable public procurement [8][9]. Even though such guidelines are followed by a number of government agencies worldwide, we are not aware of studies on software engineering to incorporate these guidelines into public companies’ procurement systems. In the private sector, companies are increasingly aware of the advantages of maintaining an “green” public image [4]. Procurement systems offer such an opportunity, as it allows companies to publicize a sustainability-related initiative. Also, by using sustainable raw materials, companies may eventually advertise so directly in the label of their products. Finally, sustainable procurement can bring financial gains [9][4].

In this paper, we report on the experience of modelling sustainability into the procurement system of one of the leading energy companies in the world. Due to a non-disclosure agreement, the anonymity of the company must be preserved; thus, a fictional name is used: Oil.Br. The system supports the process of purchasing products and services for part of the company in Brazil, including the request, approval, quotation, order submission to suppliers. The system aims to ease the acquisition of products and services, while ensuring that Oil.Br’s suppliers meet compliance rules. It also optimizes for price, payment conditions and delivery times. This paper

describes the experience of adding the sustainability dimension to this optimization criteria. Our driving question is: "Is it possible to make procurement systems more sustainable? If so, how can this be made?". Although this answer is addressed in the context of Oil.Br, the experience described in this paper can be valuable to other organizations, in both private and public sectors, wishing to implement procurement systems (or to augment their systems) with sustainability.

Our proposition was to adapt the guidelines for sustainable public procurement [8] [9] to Oil.Br's system, which raised an important concern: How to ensure the incorporated guidelines are aligned with Oil.Br's business goals and yet create a shared responsibility for sustainability within the company? In order to choose the technique to model sustainability into Oil.Br system, we started from the premise that sustainability can be considered as a software quality matter [10], so we focused on techniques that treated non-functional requirements as first-class entities, such as goal modelling. In this context, we opted for the KAOS framework [11]. As it is discussed later, the advantages of dealing with sustainability using goal modelling include the alignment with high-level business goals, the trade-off analysis, and the distribution of responsibility among agents. The methodology for the study consisted of (i) interviewing stakeholders at Oil.Br, (ii) building a preliminary model using the KAOS framework and the heuristics in [11], (iii) studying the literature on sustainability-related norms and guidelines for sustainable public procurement, (iv) incorporating sustainability goals into the system, and (v) evolving the model based on stakeholder feedback.

The contribution of the paper is two-fold. First, it strengthens the body of knowledge on sustainability and software engineering, by describing and analysing the application of a requirements engineering technique to incorporate sustainability to a large real-world system. Such need was recognized by other authors [12][13][14][15]. Second, the reported experience and the created model can help other organizations to implement sustainable procurement systems. The paper also highlights the advantages and challenges of introducing sustainability goals into a private procurement system.

This paper is organized as follows: Section I describes how IT, in general, and procurement systems, in particular, can contribute to sustainable development. Section II introduces the key concepts underlying this work, namely sustainability, sustainable public procurement guidelines, and KAOS. Section III presents the real-world procurement system to which sustainability should be incorporated. Section IV describes how this system was modelled, include an adaptation of the guidelines for sustainable public procurement. Section V shows a critical evaluation of this experience. Finally, Section VI reports on related work, and Section VII highlights the main points of this paper and lists future work.

II. BACKGROUND

A. Sustainability

Sustainable development rests on three pillars: social, economic and environmental. Social sustainability includes human

rights, as well as rights of justice, labour, equality, diversity, health, democracy and governance. It refers to a set of actions to improve the quality of life of our society [1][16]. Economic sustainability refers to the economic growth and productive efficiency of public and private sectors, including the profit that companies get, in a sustainable manner, through the distribution and consumption of goods and services [1][17]. Environmental sustainability refers to meeting the needs of the society without exceeding the capacity of ecosystems nor diminishing biological diversity [18].

Like on many other areas, Information and Communications Technology (ICT) has an impact on sustainable development. On one hand, it can implement and monitor sustainability strategies, raise awareness of the impact of actions, or even augment ordinary systems so that they achieve their business goals in a more sustainable manner. On the other hand, ICT can negatively impact sustainability by increasing energy consumption and electronic waste disposal or by creating unexpected rebound effects, for example. With respect to the environmental impact, Berkhout and Hertin [19] define the following three orders of effects caused by ICT:

First order impact: Environmental effects directly caused by the production and use of ICT, such as the resource usage and the pollution associated with the production of infrastructure and devices, power consumption of hardware, and the collection of electronics disposal;

Second order impact: Environmental impact indirectly related to the effect of ICT on the economy, production processes, products and distribution systems;

Third order impact: Indirect effects on the environment, mainly through increasing consumption and economic growth caused by the adoption of ICT.

Even though Berkhout and Hertin were concerned with environmental impact, these orders also could be thought in the context of social and economic impact.

B. Sustainable Public Procurement

Government expenditure can represent between 15% and 30% of the Gross Domestic Product [20]. Sustainable procurement aims to incorporate aspects to reduce social and environmental impacts into the process for purchasing goods and hiring services, while generating savings to the public administration [8][9]. This solution integrates social and environmental considerations through actions such as buying what is really necessary and making decisions based on the circumstances in which a product was made.

Sustainable procurement can bring many benefits. For instance, by choosing a product that has low power usage and low water consumption, for example, organizations can significantly reduce their costs during the product life cycle [9]. A key point of sustainable procurement is that it cannot result in significant additional expenditure [8].

Besides profitability, a sustainable procurement policy can bring a number of benefits not only to organizations, but also to the government and the society in general [8][9]:

- Improve organization's image, given that caring about the environment normally leads organizations be better regarded by the society and government agencies;
- Improve local community's quality of life, e.g. through the use more energy-efficient and less polluting means of transport;
- Contribute to global sustainability, by reducing greenhouse gases and deforestation in regions beyond the location
- Encourage suppliers to innovate products and to make them more sustainable, promoting a greener and more inclusive economy and also increasing the number of jobs;
- Improve organizational efficiency, allowing better decision making on procurement and contracting;
- Achieve higher levels of sustainability with the same capital due to the offset of costs during the products life cycle.

Among the different methodologies that help the government agencies to implement sustainable procurement, the best known and most used one is ICLEI's Procura+ [8][9]. This methodology defines the following important guidelines:

Watch continuous improvement: The decision criteria of purchases and contracts should go beyond price, time and quality, also assessing aspects such as the replacement of polluting sources, waste reduction and recycling, water and energy savings, combating of slave labour, social inclusion and improved relationship with communities;

Watch the product: Evaluate the product life cycle, analysing the money spent and the social-environmental impacts arising from the acquisition, use, maintenance, transportation and proper disposal;

Watch the buyer: Purchasing staff must be aware of the company's sustainability strategy and must be trained on the sustainability criteria and best practices. Furthermore, buyers need the support of an sustainability expert;

Watch the supplier: Criteria of sustainability in purchases and contracting must be checked. Dialogue must be established with suppliers to exchange ideas and receive feedback.

According to Biderman et al. [9], even though sustainable procurement is normally adopted by government agencies, it can also be adapted to the private sector, bringing benefits such as the ones mentioned previously.

In order to perform this work, we have also investigated the ISO 14000 family, which defines guidelines for environmental management [21], and some models on sustainability indicators [22] [23]. Because these addressed sustainability in general, adapting the guidelines for sustainable public procurement [8][9] was more suitable to the characteristics of the system and needs of the organization.

C. KAOS

KAOS is a goal-oriented framework for requirements engineering. Its main model, the goal model, is concerned with "why" the system needs a given functionality and "how" this functionality can be obtained. We chose the KAOS modelling because the goal model fits the needs of sharing responsibility for sustainability and aligning to high-level goals. Another influencing factor for this choice is that sustainability can

be considered as a software quality matter [10], and other techniques such as user stories and use cases are not intended to handle non-functional requirements.

The goal model's main concepts are: goals, agents, requirements, expectations and domain assumptions [11]. Goals may be formulated at different levels of abstractions, ranging from high-level strategic concerns to low-level technical concerns. Goals may also be functional, describing the services to be provided by the system, or non-functional, associated with quality of service. In a goal model, goals are interrelated by AND- and OR-refinements. An agent is an active system component that is responsible for satisfying a goal. Agents may belong to the system environment or to the software system itself. Goals should be refined until they can be assigned to a single agent. A requirement is a goal under the responsibility of a single agent in the system-to-be, while an expectation is a goal under the responsibility of an agent in the environment. Finally, a domain assumption is a descriptive statement about the environment that is expected to hold true regardless of the system behaviour.

III. OIL.BR'S PROCUREMENT SYSTEM

Oil.Br uses an in-house developed procurement system to purchase goods and to contract services. This is a large system, implemented mostly using Oracle technology. In this paper, we refer to such system as *IProc - Integrated Procurement System*. IProc aims to reduce costs, to ease the process of purchasing and contracting, and to ensure compliance rules.

In order to hire services or buy products, these must be previously registered in the system. Examples of services are maintenance, painting, health and security services. Products, in turn, are classified direct or indirect. The former are used directly in the products sold by Oil.Br; examples include base oils, additives, and packaging for lubricants. The latter are all other products used by Oil.Br's employees during processes, such as computers, paper and pens.

Both products and services must be supplied by registered companies who have passed Oil.Br's approval process, which normally consists of attesting that the supplier is reliable and has the capacity to provide what is being requested. However, there are services and goods that require certain certifications from the supplier, such as environmental management (ISO 14001 [24]) and occupational health and safety (OHSAS 18001 [25]). Examples of these are the services that require specialized equipment, and products that require especial care for waste disposal. In such cases, the corresponded certification is also verified in the approval process.

The IProc system assumes that products, services and suppliers have been previously registered. For convenience, we refer to both the purchasing of products and the hiring of services as the purchasing of items.

Oil.Br's procurement system encompasses three types of purchases: ordinary, directed and urgent. The first represents the main purchasing process. In the direct purchase, there is a pre-selected supplier this is justifiable when there is only one licensed supplier of an item. The urgent purchase is a quicker

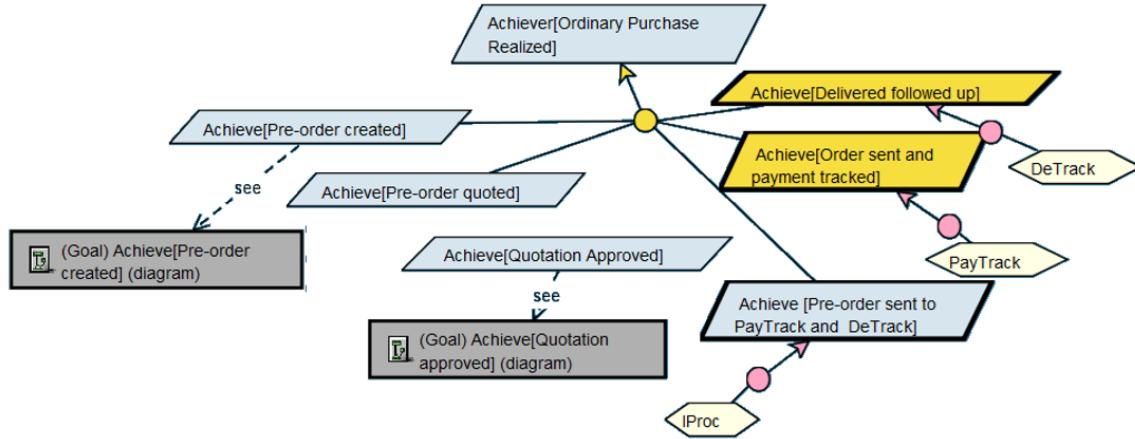


Fig. 1. Excerpt of the goal model for Ordinary Purchase.

process due to an urgent need for a product or a service, such as the replacement of a broken production equipment. The model includes the three types of purchases, but due to space constraints, only the first one is discussed in this paper.

In summary, the ordinary purchase starts when an employee (requester) asks for items, which must be approved by his assigned superior (approver). Upon approval, the system creates a pre-order and send it to the registered suppliers for quotation. Then, suppliers inform the prices of the requested items, payment conditions and delivery times. After the quotation, a supplier is chosen and the system sends the pre-order to two external systems, referred to as *PayTrack - Order Payment Tracking System* and *DeTrack - Order Delivery Tracking System*. PayTrack is responsible for issuing the purchase order to the supplier, as well as following up its payment. DeTrack controls the delivery of the product or the execution of the service. Requested items can be cancelled at any time.

In addition to the presented process, the IPProc system has three other aspects of interest: (i) it keeps track of all purchases, including information of who ordered or approved a particular item and which items were associated with the same purchase; (ii) it ensures that all requesters and approvers are registered in the system, along with their respective restrictions, such as the family of items that an employee can request or approve; and (iii) it allows users to request modifications to the system on the IT team, subject to approval.

IV. INCORPORATION SUSTAINABILITY

As described in Section I, after interviewing stakeholders, a goal model was created to Oil.Br's system. The subsequent step was to extend that model to include sustainability goals. The extended model contained 13 agents and 137 goals, from which 43 were requirements and 41 were expectations.

A. Modelling the system-as-is

A list of the agents in the system-as-is model and their main responsibilities are given as follows:

- IPProc: Oil.Br's software system to pre-order products and services (items);
- PayTrack: Oil.Br's software system to issue orders and track their payment;
- DeTrack: Oil.Br's software system to track the delivery of products and execution of services ordered;
- Requester: Employee requesting an item;
- Approver: Employee approving a request;
- Supplier: A registered provider for an item.

One of the main high-level goals of the system is *Achieve[Purchase process automated]*, which contributes to the top-level goals *Minimize[Purchase time]*, *Maintain[Compliance with regulation]* and *Improve[Ease of the purchase process]*. In order to satisfy the goal *Achieve[Purchase process automated]*, the system must allow purchases to be made and cancelled, as well as the history of all purchases. As explained before, there are three types of purchase processes, represented by the goals *Achieve[Ordinary purchase realized]*, *Achieve[Directed purchase realized]* and *Achieve[Urgent purchase realized]*. Due to space constraints, the full goal model is not shown. The most important excerpts of this model are presented as follows.

Figure 1 shows the refinement of the goal *Achieve[Ordinary purchase realized]*. To satisfy this goal, a pre-order for the item is created and sent to the suppliers for quotation. Once the quotation is approved, the pre-order is forwarded to the external systems PayTrack and DeTrack for ordering, payment and delivery monitoring. This process is represented by the goals *Achieve[Pre-order created]*, *Achieve[Pre-order quoted]* and *Achieve[Quotation approved]*, the requirement *Achieve[Pre-order sent to PayTrack and DeTrack]*, and the expectations *Achieve[Order sent and payment tracked]* and *Achieve[Delivered followed up]*. The first three are further refined.

B. Extending the model

In the extended model, three new agents were included:

- **Ergonomic System:** Oil.Br's system responsible for the well-being of employees using the company's software;
- **EnergyMo:** Adaptation of an Oil.Br's software for monitoring the used of energy by the companies IT systems;
- **Sustainability team:** Oil.Br's employees responsible for creating and evaluating the sustainability criteria of products, services and suppliers¹.

Sustainability goals aim to minimize the three impact orders of the system, defined in Section II-A². The first impact order refers to the effects directly caused by the production and use of the system. Two opportunities to reduce this impact were identified and are represented by the goals *Achieve [Energy consumption monitoring]* and *Achieve [User health preserved]*. These are explained later. Second and third impact orders are mitigated by adapting the ICLEI Procura+ methodology [8] to Oil.Br's context.

This adaptation includes an analysis of the products, services and suppliers with respect to social, economic and environmental issues. To this end, the system adopts the concept of "sustainability levels" for products, services and suppliers, calculated from sustainability criteria and their respective weights. This classification is supported by the system, but achieved in a gradual manner with the help of a sustainability team, as it is explained later. Information obtained from such classifications are taken into account in the sub-processes for registering items, requesting items and approving quotations. Note that the goals associated with these sub-processes are represented in the model by independent sub-trees whose roots are the goals *Achieve [Items registered]*, *Achieve[Items requested]* and *Achieve[Quotation approved]*, respectively. Figure 2 shows an excerpt of the model resulting from this refinement. Each of these sub-trees is described as follows.

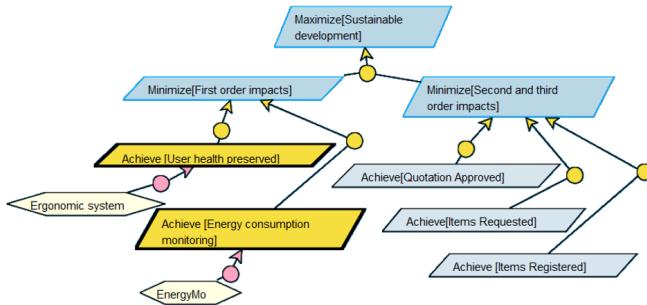


Fig. 2. Excerpt of the goal model for minimizing the three order impacts.

1) *Registering items:* This allows sustainability information to be registered with items. In the extended model, the system supports the registration of sustainability information for three groups of items: most requested items, direct items (which are used directly in the products sold by Oil.Br), and critical items (with high impact on sustainable development). The first group

is automatically recognized by the system. The latter two must be informed when registering the item.

Figure 3 shows the goals associated with this sub-process. An employee registers a product or a service and informs whether this item is critical to sustainability or directly used in Oil.Br's products; this is represented by the goal *Achieve[New item informed]*. If the newly registered item is critical or direct, the system sends a request to the sustainability team asking for the corresponding sustainability information to be included, as represented by the goal *Achieve[Sustainability information requested if critical or direct item]*.

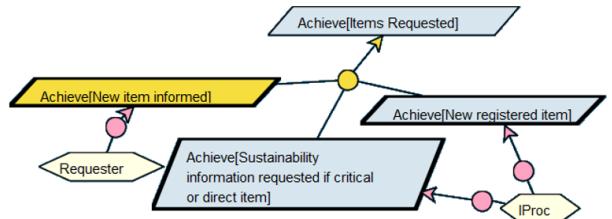


Fig. 3. Excerpt of the goal model for registering items.

2) *Requesting items:* As Oil.Br's procurement system is in production, items are already registered without any sustainability information. When requesting items, there are two sustainability-related opportunities: the first is to identify direct and critical items without registered sustainability information and request it to the sustainability team. The second is to consider the available sustainability levels and choose among alternative items. These opportunities are represented by the goals *Achieve[Sustainability information of critical or direct item requested, if not registered]* and *Achieve[Sustainability information of item available, if registered]*. Also, periodically, the system checks the purchase history, identifying the most requested items; this is represented by the goal *Achieve[Sustainability information of most requested items checked]*. If an often requested item does not have sustainability information associated to it, the system informs to the sustainability team. This excerpt is shown in Figure 4.

The refinement of the goal *Achieve[Sustainability information of the item registered]* consists in registering the sustainability criteria with their relative weights and values. These are defined by the sustainability team, so that the system can calculate the sustainability level of products and services. This refinement is shown in Figure 5.

Finally, the request for an item must be approved. Figure 6 shows the refinement of the goal *Achieve[Request responded]*. In order to contribute to sustainable development, one must ensure that only the necessary items are being ordered. Therefore, the external system DeTrack informs the amount of that item in stock and this information is made available in the system, as represented by the expectation *Achieve[Amount of item in stock informed]* and the requirement *Achieve[Amount of item available]*. This infor-

¹A compliance monitoring department already exists within Oil.Br.

²In this work, we also considered the social and economical impacts.

mation should be taken into account by the approver when verifying that an item is actually needed.

It is important to note that the system does not force a requester to choose the most sustainable item; it only shows the sustainability levels of alternative items, leaving the decision to the requester's discretion. For this reason, it is necessary to provide training to the staff, so that everyone can be aware of the company's goals with respect to sustainable procurement, being able to judge if a given purchase is really necessary, and understand the impact of their choices.

3) Approving Quotations: Once requested, items are grouped together and sent for quotation and approval. In this sub-process, suppliers respond to a letter of invitation and the approver compares quotations from different suppliers, choosing from whom to purchase. The approver should take into account the sustainability level of the quotation, in addition to the usual criteria of price, payment condition and delivery time. The sustainability level of a quotation is calculated from the sustainability levels of the supplier and items, the means of transport to delivery the items, and the type of packaging. The latter two are important because items may have sustainable features, but the transport used for its delivery or packaging may generate a high environmental impact. Likewise, a vendor may have multiple warehouses at different locations, which can alter the sustainability level of a given request.

The price is also considered. The system compares the price of a more sustainable item with the purchase history of alternative items, warning the approver when a request generates a significant increase in costs for Oil.Br.

The goals associated with the quotation approval are shown in Figures 7 and 8. In Figure 7, a better sustainability is ensured by the goals *Achieve[Purchase history informed]* and *Achieve[Purchase option chosen]*. Figure 8 shows the refinement of the goal *Achieve[Sustainability level of quotation available]*. The information about means transport and packaging are provided by the expectation *Achieve [Means of transport and packaging informed]*. The supplier's sustainability information is represented by the goal *Achieve[Sustainability level of supplier available, when registered]*. The refinement of this goal is analogous to the calculation of the sustainability level of an item and is not shown due to space constraints.

4) User Health and Resource Monitoring: The goals presented so far contribute to mitigating second and third order impacts on sustainability. Figure 2 shows the two expectations defined for minimizing the first order impact of the system.

In the expectation *Achieve[User health preserved]*, the external Ergonomic system monitors the time of continuous system usage and the amount of keystrokes, temporarily blocking the system when these become excessive. This system also shows, periodically, a reminder for the user to rest. This practice is already adopted by Oil.Br.

In the expectation *Achieve[Energy consumption monitored]*, an existing monitoring system, EnergyMo, can be adapted to inform the IT and sustainability teams

when energy usage is overcoming certain thresholds, so that improvements can be discussed and implemented.

C. Adoption Process

The effective adoption of sustainable procurement requires sustainability measures to be adopted in a gradual and continuous manner [8][9]. For this reason, in this study, a gradual adoption process composed of the following steps has been adapted from sustainable procurement guidelines: (1) map the purchase profile, (2) allocate the sustainability team, (3) develop an action plan, (4) select products and suppliers, (5) search for alternative products and suppliers, (6) provide training to staff, (7) verify and evaluate the system. The detailing of this process is out of the scope of this paper.

D. Procura+ Guidelines

The goals listed previously follow the guidelines of the ICLEI Procura+ summarized in Section II-B, as follows:

Watch continuous improvement concerns with incorporating sustainability into the decision criteria. The extended model adopts the concept of "sustainability levels" for items and suppliers, which are automatically calculated by the system based on criteria's values and respective weights. This information is gradually registered in the system by the sustainability team. IPProc uses these levels to make recommendations regarding more sustainable items and quotations, ensuring that the suggested option does not increase significantly the costs for Oil.Br.

Watch the product evaluates the sustainability of an item throughout its life cycle. This is done gradually by the sustainability team, who is advised of this need when items are being registered or requested. According to the Procura+ guidelines, an effective evaluation should consider only few criteria [8]. The sustainability team must be trained in order to choose the criteria and their relative weight.

Watch the buyer is concerned with the training of purchasing staff. Training is defined in the adoption process. It creates awareness of Oil.Br's sustainability strategy and helps employees in identifying priority items, requesting more sustainable items and approving more sustainable quotations.

Watch the supplier refers to checking whether the sustainability criteria are being met and establishing a dialogue with suppliers. In the adoption process, the sustainability team gathers information about suppliers, making them aware of Oil.Br's commitment to sustainability. That encourages suppliers to compete on sustainable practices. Note that the criteria to evaluate the sustainability of suppliers should consider the three pillars of sustainability. This ensures, for example, that their products have minimal impact on the environment, follows health and safety standards, and so on. Information regarding items and suppliers practices must be checked periodically. This is already a practice within Oil.Br for suppliers that are required to comply with standards.

V. CRITICAL EVALUATION

This was the first attempt to model sustainability in Oil.Br's procurement system.

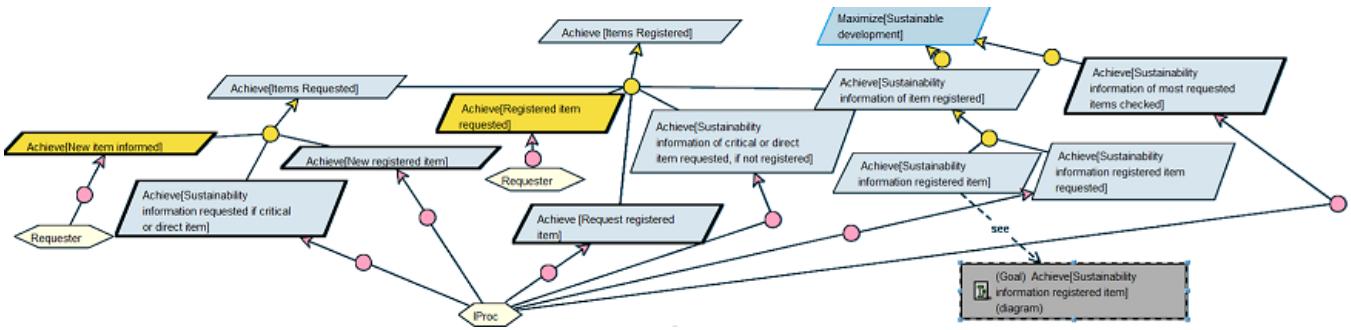


Fig. 4. Excerpt of the goal model for requesting items.

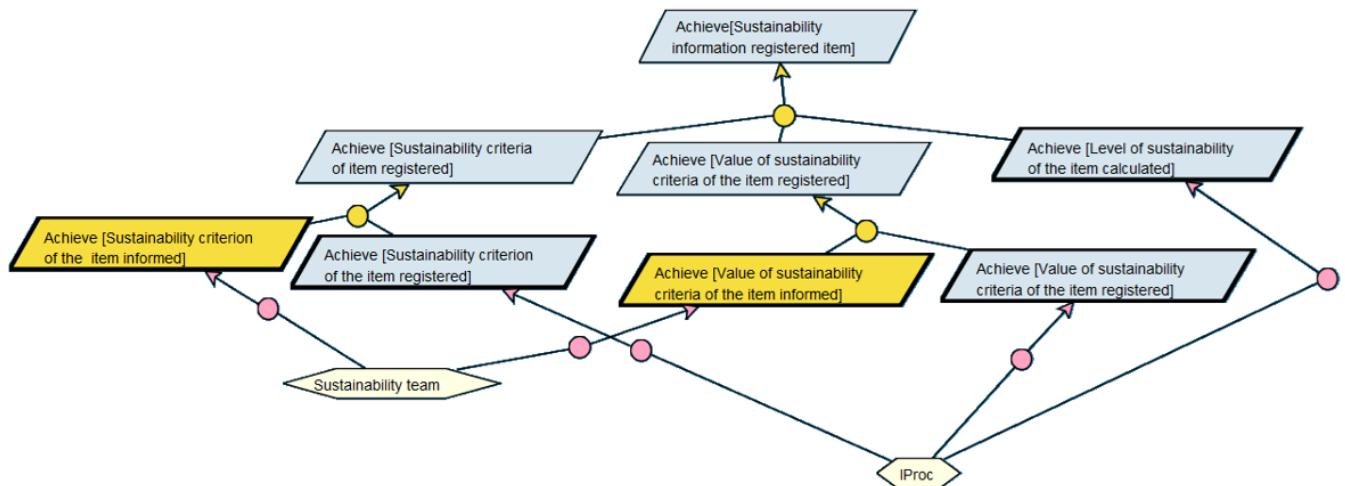


Fig. 5. Excerpt of the goal model for registering sustainability information.

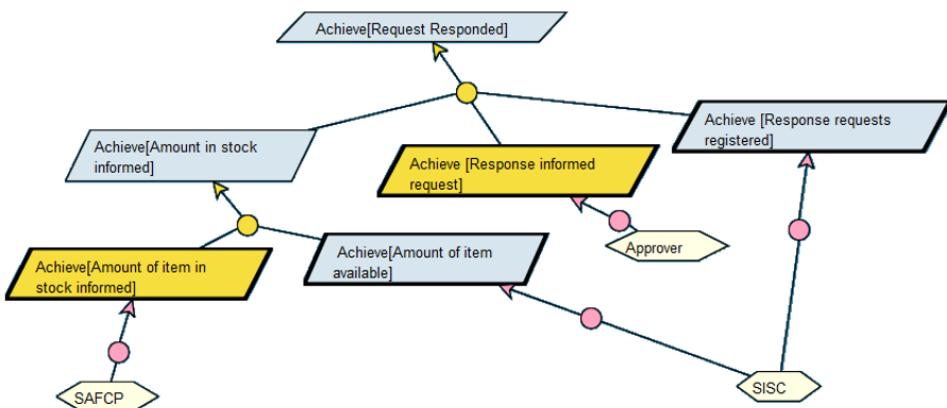


Fig. 6. Excerpt of the goal model for responding to item requests.

This involved an IT manager, a sustainability analyst, and three researchers. One researcher was experienced in KAOS and another worked for Oil.Br. Researchers conducted the interviews to build an initial model, which was then discussed and refined with the IT manager and the sustainability analyst. This section discusses the benefits and challenges identified with our experience.

A. On Incorporating Sustainability into Procurement Systems

This work aimed to respond the posed question “Is it possible to make procurement systems more sustainable? If so, how can this be made?”. Our experience has shown that it was indeed possible to do so. The approach used was goal modelling. We believe the resulting model and the experience here described can serve as a starting point to other companies wishing to implement sustainable procurement systems.

The feedback from Oil.Br’s stakeholders was positive, showing interest to take the work forward. During the study, the model also led to interesting discussions within Oil.Br, such as: How their processes could be adapted to implement the model? How sustainability goals impact Oil.Br’s business goals? Which opportunities the purchase of more sustainable raw materials can bring into Oil.Br’s final products?

Nevertheless, in order to effectively adopt the proposed model, some challenges remain. They also apply to any organizations wishing to implement such systems. They are:

Return on Investment: Although studies demonstrate that organizations with sustainable practices show financial gains [4][8], in order to promote the idea of sustainable procurement, its financial benefits must be studied. Further studies should answer questions such as: Which financial benefits such a system can bring? When these benefits will start to pay off, considering the costs to build or adapt the system? This kind of analysis is already a practice at Oil.Br.

Minimal impact on workload: Another concern is that the system should cause minimal impact on employee’s workload. This is already a concern for the ICLEI’s Procura+[8], in which our model is based. Nevertheless, in the case of Oil.Br, it is possible to reduce even more the impact on the workload, by delegating to the supplier the responsibility of providing the information on the sustainability criteria for products, services and itself. This is possible because Oil.Br already have in place an auditing process for suppliers.

Sustainability Indicators: In order to implement the last step of the process described in Section IV-C, one needs to define indicators for monitoring and system improvement. In the case of Oil.Br, some indicators of interest would be the number of suggested sustainable items that have been requested, the difference in price between sustainable items and their alternatives, how many items had their sustainability information registered, whether the purchase of more sustainable direct items reflect on sales etc. Such indicators could be (at least partially) collected and analysed by the system, in order to identify improvement opportunities. This type of support has not been considered in this first modelling exercise.

B. On Goal Modelling for Sustainability

With respect to goal modelling, some of its well recognized advantages were particularly useful for incorporating sustainability into software/socio-technical systems:

Problem definition: The sustainability impact of social-technical systems goes beyond the energy spent to run the software product, affecting the three pillars of sustainability. It also includes the direct and indirect impact of use of the software product, as well as its associated processes. Goal-oriented approaches, such as KAOS, are particularly useful to reason about problems associated with business structures, processes and their support systems.

Goal refinements and trade-off analysis: Usually there are concerns and resistance about the viability of implementing sustainable practices. By refining high-level goals into low-level goals and by showing the positive and negative contribution among goals, goal modelling techniques allow for a more objective analysis of the advantages and disadvantages of incorporating sustainability into a system, including a clear discussion about the impact on high-level business goals.

Assignment of responsibility: An important aspect about sustainability is that it cannot be ensured by isolated efforts. It is a concern of the whole society, requiring the cooperation of a large number of members. The same applies to sustainability within organizations. Sustainability initiatives can be jeopardized if such a concern is ignored by key organization’s sectors. In this sense, goal modelling can be particularly useful in creating a shared responsibility for sustainability, by assigning goals to different agents.

Variables and functions: In order to evaluate whether the sustainability practices are being adopted and whether their envisioned benefits are being realized, indicators are crucial. KAOS allows for modelling quality variables and objective functions to measure the satisfaction of goals. This can be very helpful to define and monitor sustainability indicators.

Risk analysis: Finally, sustainability is closely related to risk mitigation. Many sustainability practices seek to avoid or minimize negative impacts on the natural environment and the society. Goal modelling facilitates the discussion between software engineers and sustainability experts in order to analyse risks, discuss alternative solutions and set priorities.

Besides the usual difficulties faced in any requirements engineering process, our main obstacle was the fact that two of the researchers were not experienced with the KAOS technique. This was overcome with training sessions and the support of the experienced researcher. Since the Oil.Br stakeholders have a wide background in systems engineering, an introduction to the technique was sufficient for the stakeholders to understand the models. With respect to goal modelling, some challenges may be faced in subsequent steps, such as the following:

On monitoring sustainability indicators: Sustainability is, by nature, a subjective concept. Can quantitative variables on goals alone be used as indicators of sustainability? How to objectively combine variables spread among goals? How to model variables and functions whose values and targets may

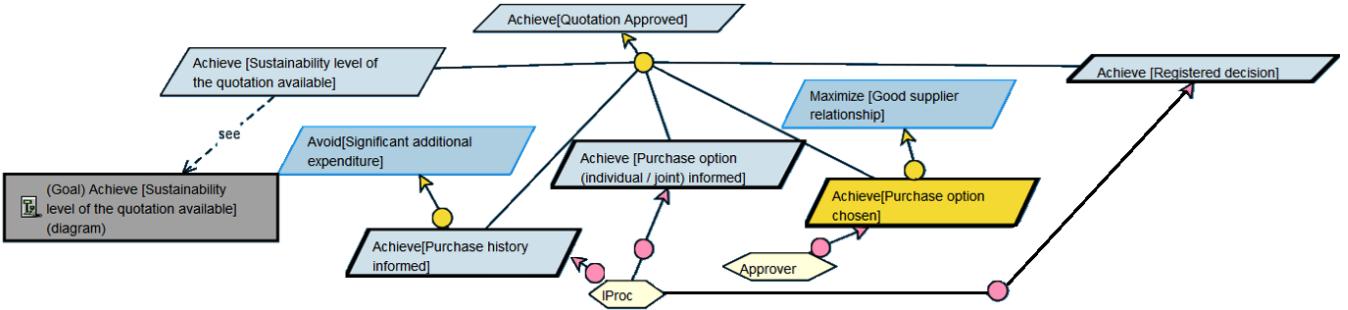


Fig. 7. Excerpt of the goal model for approving quotations.

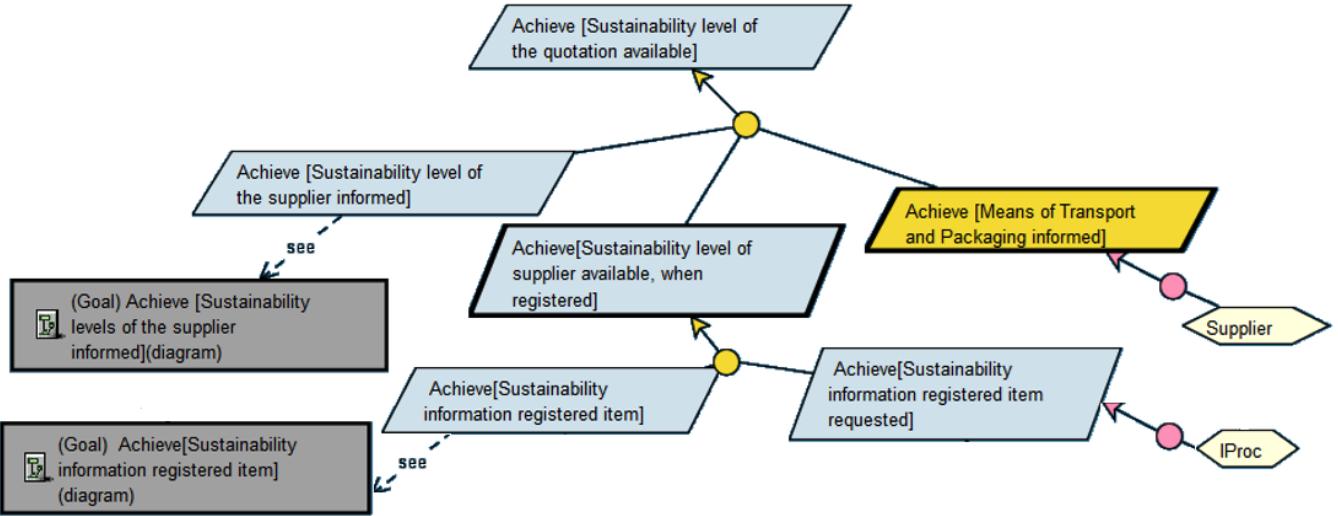


Fig. 8. Excerpt of the goal model for defining quotations sustainability levels.

vary over time? A related problem has been identified by Stefan et al. [26], concerning how alternative designs can be analysed in the face of time-varying variables.

On financial analysis: The financial benefits of sustainability may be difficult to quantify. In KAOS, variables and functions are defined at the goal level. For lower-level goals, some limited analyses can be made, such as the number of sustainable items bought or the money spent on sustainable items against alternatives. However, for high-level goals such as Improve[Organizations public image], the financial benefit can be difficult to analyse, not always being a simple propagation of values from lower-level goals.

VI. RELATED WORK

Other works have used goal modelling to reason about sustainability. Most of them, like ours, created models for specific domains. In [27], a software application to aid users in the development of a sustainable backyard food and resource system was modelled. The problem of modelling sustainability for conference organization is tackled in [15] and [28]. Stefan et al. [26] applied KAOS to model sustainability goals of a large university in central London. In [12], the authors examined different Requirements Engineering techniques to model sustainability as a first-class entity. They used goal

analysis to determine priorities for sustainability goals of a software tool to support the organization of events. Generally, authors of these papers found goal-oriented requirements engineering to be a suitable approach to model sustainability.

Generic models and taxonomies for sustainability have also been proposed. Penzenstadler and Femmer [29] created a generic sustainability model that can be instantiated for products and processes. This model has been used for particular domains, as in [28]. Cabot et al. [15] state that sustainability should be a softgoal and present a taxonomy of sustainability-related goals. Nevertheless, as recognized by the authors, the taxonomy was driven by the needs of their case study and was not generic enough. Observing this limitation, the authors of [12] proposed a new one, but pointed out that their taxonomy was tentative and needed more research. In our study, we chose to construct our model directly, focusing on the three order impacts [19] and using internationally accepted guidelines in the domain of public procurement [8][9]. A future research can map our goals to a generic sustainability model or taxonomy, such as the ones mentioned previously, and observe whether they can help us to identify sustainability goals that might have been overlooked.

Finally, there are a number of other studies that look at requirements for procurement systems, as listed in Benslimane's survey [30]. Nevertheless, we are not aware of any work that deals specifically with sustainability.

VII. CONCLUSION AND FUTURE WORK

This paper reports on our experience with modelling sustainability goals for a procurement system of a large energy company in Brazil. Procurement systems can have a considerable influence on the sustainable development of the communities around them. Our model focus on mitigating the three impact orders defined in [19] and was based on internationally accepted guidelines in the domain of public procurement [8][9]. We believe this experience can be valuable to other companies wishing to implement such systems.

We have observed that some of the advantages of KAOS are particularly useful for sustainability: it helps to reason about sustainability considering not only the software, but also their surrounding processes and systems; it allows a clear discussion about the impact of sustainability practices on business goals; it helps to create a shared responsibility for sustainability and to define indicators; it facilitates the discussion with domain experts to analyse sustainability risks.

The model led to interesting discussions with Oil.Br's stakeholders about the organization's business goals and opportunities. Yet, for adopting the proposed model, some challenges need to be addressed as future work, such as: analysing the return on investment of the system, studying forms of minimizing the impact on the users workload, and extending the model to support the collection and analysis of sustainability indicators. Furthermore, we plan to compare our model to existing generic models and taxonomies, in order to identify further goals and validate the generality of these proposals.

ACKNOWLEDGEMENTS

The authors would like to thank Ricardo, the IT manager at Oil.Br, Aline Matias, a sustainability analyst, and Dr. Emmanuel Letier from their valuable discussions on this work.

REFERENCES

- [1] UN World Commission on Environment and Development, "Report of the world commission on environment and development: Our common future," in *UN Conference on Environment and Development*, 1987.
- [2] B. Pernici, M. Aiello, J. vom Brocke, B. Donnellan, E. Gelenbe, and M. Kretsis, "What is can do for environmental sustainability: A report from caise11 panel on green and sustainable is," in *Communications of the Association for Information Systems*, vol. 30, 2012.
- [3] M. F. Gasparino and M. de Souza Ribeiro, "Análise de relatórios de sustentabilidade, com ênfase na GRI: Comparação entre empresas do setor de papel e celulose dos EUA e Brasil," *Revista de Gestão Social e Ambiental*, vol. 1, no. 1, 2007.
- [4] E. Arantes, "Investimento em responsabilidade social e sua relação com o desempenho econômico das empresas," *Conhecimento Interativo*, vol. 2, pp. 3–9, January/June 2006.
- [5] P. Lacy, T. Cooper, R. Hayward, and L. Neuberger, "A new era of sustainability: Un global compact-accenture ceo study," Tech. Rep., 2010.
- [6] ———, "The un global compact-accenture ceo study on sustainability: Architects of a better world," Tech. Rep., 2013.
- [7] B. Tomlinson, *Greening Through IT: Information Technology for Environmental Sustainability*. The MIT Press, 2010.
- [8] S. Clement, C. Erdmenger, T. Held, R. Barth, I. Oehme, R. Pierrard, B. Lackner, and V. Fuhr, "The Procura+ Manual: A guide to cost-effective sustainable public procurement," Tech. Rep., 2007.
- [9] R. Biderman, L. S. V. d. Macedo, M. Monzoni, and R. Mazon, "Guia de compras públicas sustentáveis: uso do poder de compra do governo para a promoção do desenvolvimento sustentável," Tech. Rep., 2008.
- [10] C. Calero, M. F. Bertoia, and M. Á. Moraga, "Sustainability and quality: Icing on the cake," in *RE4SuSy@ RE*. Citeseer, 2013.
- [11] A. van Lamsweerde, *Systematic Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons, 2008.
- [12] M. Mahaux, P. Heymans, and G. Saval, "Discovering sustainability requirements: An experience report," in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6606, pp. 19–33.
- [13] N. Amsel, Z. Ibrahim, A. Malik, and B. Tomlinson, "Toward sustainable software engineering (NIER) track," in *33rd International Conference on Software Engineering*. ACM, 2011, pp. 976–979.
- [14] B. Penzenstadler, A. Raturi, D. Richardson, C. Calero, H. Femmer, and X. Franch, "Systematic mapping study on software engineering for sustainability (SE4S) protocol and results," Institute for Software Research, University of California, Tech. Rep. UCI-ISR-14-1, 2014.
- [15] J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, and J. Mazn, "Integrating sustainability in decision-making processes: A modelling strategy," in *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, 2009, pp. 207–210.
- [16] S. McKenzie, "Social sustainability: Towards some definitions," ser. Hawke Research Institute Working Paper Series 27, 2004.
- [17] E. Michels, P. E. A. Grijó, E. Machado, and P. M. Selig, "Knowledge management, project integration and corporate sustainability: is there a link?" *Journal of Project, Program & Portfolio Management*, vol. 3, no. 2, pp. 17–27, 2013.
- [18] J. Morelli, "Environmental sustainability: A definition for environmental professionals," *Journal of Environmental Sustainability*, vol. 1, pp. 19–27, 2011.
- [19] J. H. Berkhout, F., "Impacts of information and communication technologies on environmental sustainability: Speculations and evidence report to the oecd, p.1," Tech. Rep., 2001.
- [20] Secretaria da Administração ICLEI-Brasil, "Compras públicas sustentáveis: uma abordagem prática," Tech. Rep., 2012.
- [21] C.-C. Chen, "Incorporating green purchasing into the frame of iso 14000," *Journal of Cleaner Production*, vol. 13, no. 9, pp. 927–933, 2005.
- [22] F. Tayra and H. Ribeiro, "Sustainability indicators models: synthesis and critical evaluation of the main experiences," *Saúde e Sociedade*, vol. 15, no. 1, pp. 84–95, 2006.
- [23] R. M. T. Santos, "Compras públicas sustentáveis-a utilização do poder de compra do estado no fomento de produtos ecologicamente corretos na fiocruz," Ph.D. dissertation, Escola Nacional de Saúde Pública Sergio Arouca, 2011.
- [24] "ISO 14004.(2004)," *Environmental management systems-General guidelines on principles, systems and support techniques*.
- [25] "OHSAS 18001: 2007," *Occupational health and safety management systems*. London, 2007.
- [26] D. Stefan, E. Letier, M. Barrett, and M. Stella-Sawicki, "Goal-oriented system modelling for managing environmental sustainability," in *WSRCC-3 meeting notes*, 2011.
- [27] J. Norton, A. J. Stringfellow, J. J. L. Jr., B. Penzenstadler, and B. Tomlinson, "Plant guild composer: A software system for sustainability," in *RE4SuSy@ RE*, ser. CEUR Workshop Proceedings, vol. 995, 2013.
- [28] B. Penzenstadler and H. Femmer, "Re@21: Time to sustain!" ser. CEUR Workshop Proceedings, vol. 995. CEUR-WS.org, 2013.
- [29] ———, "A generic model for sustainability with process- and product-specific instances," in *Proceedings of the 2013 Workshop on Green in/by Software Engineering*, ser. GIBSE '13, 2013, pp. 3–8.
- [30] Y. Benslimane, L. M. Cysneiros, and B. Bahli, "Assessing critical functional and non-functional requirements for web-based procurement systems: A comprehensive survey," *Requir. Eng.*, vol. 12, no. 3, pp. 191–198, Jul. 2007.

A Case Study using a Protocol to Derive Safety Functional Requirements from Fault Tree Analysis

Luiz Eduardo Galvão Martins

Institute of Science and Technology

Federal University of São Paulo, UNIFESP

São José dos Campos, Brazil

legmartins@unifesp.br

Tiago de Oliveira

Institute of Science and Technology

Federal University of São Paulo, UNIFESP

São José dos Campos, Brazil

tiago.oliveira@unifesp.br

Abstract— State-of-the-art in Requirements Engineering offers many frameworks and techniques to enable requirements engineers in their work. However, for critical systems there are gaps in state-of-the-art, and these can result in dire consequences, potentially putting lives in danger and damage infrastructure and threaten the environment. A well known technique used to help requirements engineers to understand safety hazards situations in the context of safety-critical software is Fault Tree Analysis (FTA). This technique is a good one to decompose hazards identified in the system context into events that may put the system functionalities in risk. However, FTA does not offer a protocol of how to derive safety functional requirements from fault trees. In this paper we present a case study adopting a protocol to help requirements engineers to derive safety functional requirements from FTA. The proposed protocol was based on a study performed in a Brazilian company in the area of electronic medical devices. The development of prototype of a low cost insulin infusion pump, which is a critical system, offered the basis to propose and test a protocol to derive safety functional requirements from FTA. During the case study we collected evidences that help us to discuss if FTA is sufficient to guide software engineers to implement the corresponding control software and also if FTA offers enough information to help requirements engineers to derive safety functional requirements.

Index Terms— Safety Functional Requirements, Fault Tree Analysis, Critical Systems, Hazard Situations, Embedded Systems.

I. INTRODUCTION

Modern society is becoming more and more dependent on software. In the last decades there were important advances in computing technologies, and software assumed a central role in the control of equipments, devices and processes based on computers. On a daily basis, the common citizen is dependent on several services supported by software, and many services are transparent to the users. For example, a car contains millions lines of code, that control everything from the A/C unit to safety critical aspects such as breaking. Other examples range from banking services available in automated teller machines or web systems; services available in pay-per-view TVs; traffic-air control; road monitoring; telecommunication system control; medical equipments control, since a blood pressure monitor device until sophisticated magnetic resonance image equipments; automotive control systems; entertainment

systems; to surveillance systems based on computer vision; and others [2][3].

Research in Software Engineering is related to the conception, design, development and evolution of software-intensive systems [4]. Furthermore, Software Engineering is becoming specialized in many fields, considering the scope and the complexity of several aspects concerned to software development, such as Requirements Engineering, which is also considered a field of Systems Engineering.

Requirements Engineering focuses on the development of processes, techniques, methods, models and tools to help in the conception of software and systems, covering the activities of elicitation, analysis, specification, validation and management of requirements [4]. Requirements are the base to further software design, but besides this it offers a description of the functional and non-functional aspects that must guide the implementation and evolution of the software, as well as the verification and validation activities central for safety and security.

The software development teams are concerned with the development of systems that address user needs. The correct identification of such needs is not a trivial task, and usually there are many issues to elicit and define the relevant software requirements. Software-intensive systems are becoming ubiquitous in daily life, making people highly dependent on them. Thus, systems must offer a high level of safety, security, reliability and robustness [5][11]. However, despite the efforts of development teams, the level of customer satisfaction and quality are often lacking [1][4]. This is both due to inadequacies in the ability to elicit and select the central and most important requirements to realize in the product, but also due to the fact that requirements understanding leads to miscommunication within the development companies.

State-of-the-art in Requirements Engineering offers many frameworks and techniques [4][12][14] to enable requirements engineers in their work. However, for critical systems there are gaps in state-of-the-art, and these can result in dire consequences, potentially putting lives in danger and damage infrastructure and threaten the environment [1][6][9].

A well known technique used to help requirements engineers understand hazards situations in the context of safety-critical software is Fault Tree Analysis (FTA) [18]. This technique is a good one to decompose hazards identified in the system

context into events that may put the system functionalities in risk. However, FTA does not offer a protocol of how to derive safety functional requirements from fault trees. Despite the fault trees are useful to represent events that put systems in hazardous states they do not describe safety functional requirements and only the graphics representation supported by FTA seems to be insufficient to document safety requirements.

In this paper we present a case study adopting a protocol to help requirements engineers to derive safety functional requirements from FTA. This protocol was performed in a Brazilian company in the area of electronic medical devices to derive requirements in styles “should” and “should not”, which helped software engineers understand the safety functional requirements to be implemented in the control software of an insulin infusion pump. During the case study we collected evidences that help us to discuss if FTA is sufficient to guide software engineers to implement the corresponding control software and also if FTA offers enough information to help requirements engineers to derive safety functional requirements.

The motivation to the Brazilian company to develop a low cost insulin infusion pump is that until now there are no companies in Brazil developing such device. For this reason, diabetes patients in Brazil have to import insulin infusion pump whose price is about 5,000 USD including import tax, which is very expansive for the most of the Brazilian people.

This paper is organized as follows: section 2 presents background and related works in safety-critical software; section 3 outlines research methodology to conduct the case study based on development of a low cost insulin infusion pump in a Brazilian company; section 4 presents the results and analysis of the performed case study; and section 5 concludes the paper and points out to future works.

II. BACKGROUND AND RELATED WORKS

The main characteristic of safety-critical software, that distinguishes it from others, is that error or failure occurrence is potentially dangerous to people, properties and environment, with potential to cause accidents. For this reason, this kind of software demands strict management of safety aspects [14][16]. Table I shows complementary characteristics of safety-critical software, as well as the gaps in current requirements approaches.

According to Firesmith [1] most requirements specifications produced in industrial setting are incomplete and do not properly address hazard situations for safety-critical systems. Requirements specifications are particularly incomplete to consider exceptional situations that combine rare events sequences, culminating in unacceptable risks [19][20]. Even when requirements are specified for rare combinations of events, they may be ambiguous, incorrect or inconsistently implemented [6][10]. Therefore, safety related requirements must be carefully specified, demanding more sophisticated Requirements Engineering approaches. Fig. 1 presents a high level model showing the main system conditions taking into account the safety viewpoint in the development of safety-critical systems. The state machine highlights three main transitions to systems with potential to cause accidents. Hazard

conditions are the central issue in this model. Capturing and specifying requirements that help system engineers deeply understand all possible system hazard conditions is probably the main contribution that Requirements Engineering approaches can offer in the domain of Safety Engineering. For such intention a complete specification of the sequences of risk events, hazard mitigation events and catastrophic events must be done.

TABLE I. COMPLEMENTARY CHARACTERISTICS OF SAFETY-CRITICAL SOFTWARE AND GAPS IN CURRENT REQUIREMENTS APPROACHES.

| Characteristics of Safety-Critical Software | Gaps in Current Requirements Approaches |
|---|---|
| High integrity level. | The majority of software safety standards adopt an integrity level approach, categorizing software in terms of its criticality [12]. Current requirements approaches do not support integrity compliance properly [17]. |
| High complexity system. | Safety-critical software usually works in high complexity systems. As discussed in [13], abstraction layers facilitate reasoning about system characteristics, but existing requirements approaches are insufficient for handling requirements for critical systems. |
| High associated development cost. | Current software safety standards assume that safety is proportional to cost [12], that is because critical systems need more process and documentation. This is justified by the necessity of assurance activities during all the process. Requirements approaches that emphasize design simplicity can help to cost reducing. Current requirements approaches do not address simplicity as a key feature. |
| Risk tolerance. | Establishing risk tolerance and acceptance is a fundamental issue during safety requirements definition. Despite the safety standards give high attention for that, current requirements approaches do not properly address such issue. |

Some hazard assessment approaches have been proposed in the context of Safety Engineering. The most common are Checklists, Failure Modes and Effects Analysis (FMEA), and Fault Tree Analysis (FTA) [18]. The Checklist approach is easy to use and very common, and can be used at all phases of system project. However, Checklists are difficult to compile and easy to misuse. No matter how much effort is dedicated to create a Checklist, there is no guarantee that it is complete.

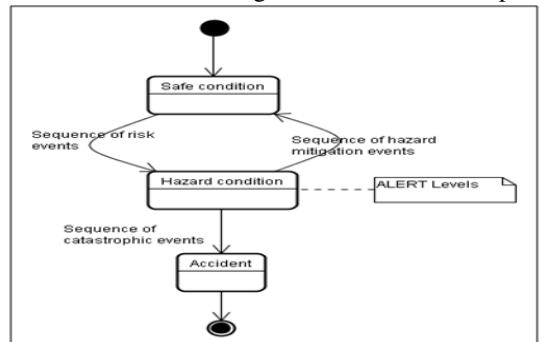


Fig. 1. High level model of system conditions from the safety viewpoint.

FMEA was originally designed for reliability issues, the goal of this approach is to analyze the possible failure modes of a component and to identify the consequences of such failures. FTA offers a structure that helps requirements engineers to analyze risk events with the intention to determine the environmental conditions under which a state system becomes unsafe. However, for FTA to become useful, a small number of critical failures must be chosen which can be distinguished from a large number of possible failures by their catastrophic consequences [18].

In [19][20] Lutz and Mikulski discuss the importance to identify rare events in safety-critical systems. They present seven cases where rare events contributed to put spacecraft's missions in serious risks. As mentioned in their works, rare events usually have the potential to cause catastrophic risks for critical system. For this reason, effort to identify rare events is strongly advised. However, the current requirements approaches are not adequate for such intention.

III. RESEARCH METHODOLOGY

The case study presented in this paper was performed during the initial phase of a project to develop a prototype of a low cost insulin infusion pump. That project has been developed in cooperation with a Brazilian company called *DeltaLife* (www.deltalife.com.br), which is a company in the area of electronic medical devices. It was identified a gap in Brazilian market that motivated industry and academia to join forces for such development.

Participated in that study six people: one mechanical engineer – with more than 5 years of industry experience; one electronic engineer – with more than 10 years of industry experience; one requirements engineer – with more than 10 years of experience in industry and academia; two software engineers with two years of industry experience; and one doctor with more than five years of experience in diabetes treatment using insulin infusion pump. Both mechanical and electronic engineers had high experience in embedded systems development. The requirements engineers and software engineers had few experience in embedded systems.

The first step of that study was to identify the main hazard situations concerned to insulin infusion pump. The sources to identify the hazards involving insulin infusion pump were: (i) literature review [7][8]; (ii) the doctor experience in the diabetes treatment using insulin infusion pump; and (iii) the mechanical and electronic engineers experience in the embedded systems development for electronic medical devices.

It was identified ten relevant hazard situations that could put the user in danger. The main hazard situation identified was “insulin overdose” that can lead the user to death. Considering that “insulin overdose” is the most severe hazard situation it was chosen to test the adopted protocol to derive safety functional requirements from FTA.

FTA is a well known technique used to analyze hazard situations in safety-critical systems, it is a technique very used in safety engineering community and it is being adopted by requirements engineers to help in the safety requirements

capturing. In that study FTA was used to decompose “insulin overdose” into risk events as presented in Fig. 2.

Considering that is becoming common requirements engineers to adopt FTA as technique to analyze hazard situations in safety-critical systems, two research questions (RQ) were investigated in the conducted case study:

- *RQ1 – Does FTA offer enough information to software engineers implement safety requirements in the corresponding control software?*
- *RQ2 – Is FTA a good enough representation to derive safety functional requirements which describe system hazard situations?*

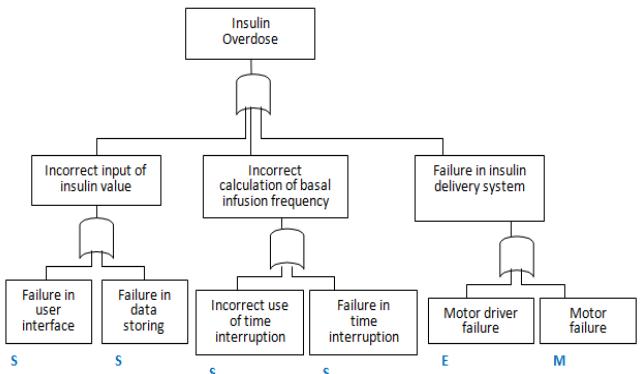


Fig. 2. Fault tree for the “insulin overdose” hazard with classification tags produced during the case study.

Fig. 2 shows the decomposition of “insulin overdose” into risk events using FTA technique. FTA is a top-down analysis starting from a hazard situation that should be decomposed into risk events. The number of decomposition levels depends on the complexity of the hazard situation analyzed. The fault tree presented in Fig. 2 is a partial view of the “insulin overdose” hazard, which was produced by the requirements engineer during the case study. We consider that it is enough to discuss the problem of safety functional requirements derivation.

The fault tree leaves represent possible causes that can put the system in hazard state. The intermediate levels of the fault tree represent groups of risk events and these levels are useful to organize the fault tree analysis. To improve the FTA understanding we put tags near the fault tree leaves. These tags identify if the failure cause is a software cause (S), an electronic cause (E), or a mechanical cause (M).

The development of a prototype of low cost insulin pump, which is a critical system, offered the basis to propose and test a protocol to derive safety functional requirements from FTA. The proposed protocol is discussed in section IV.

This protocol was performed to the “insulin overdose” hazard and 12 safety functional requirements were derived from the corresponding FTA. To validate the study a questionnaire was used which asked the difficulty to use FTA as a guide to implement safety features in the control software,

and the improvement when FTA is accompanied with “should” and/or “should not” requirements.

The questionnaire was answered by two software engineers that participated in the case study. The results and analysis of that study are discussed in the next section.

IV. RESULTS AND ANALYSIS

As stated in the previous section we chose the “insulin overdose” among ten relevant hazards identified in the context of insulin infusion pump to perform a protocol to help requirements engineer to derive safety functional requirements from FTA. The adopted protocol was divided into three steps:

1. Build FTA for hazard situation;
2. Classify fault tree leaves as software, electronic or mechanical failure causes;
3. For each failure cause (fault tree leaf) to write requirements descriptions in the style “should” and/or “should not” requirements.

A. Build FTA for hazard Situation

In the adopted protocol FTA assumed a central role to requirements engineer gets information to derive safety functional requirements. Therefore, FTA was carefully performed by the requirements engineer that participated in the case study.

The hazard that was chosen to be analyzed was directly mapped as the root in the fault tree. Starting from this point the requirements engineer used all the experience from the mechanical and electronic engineers as well the doctor experience to identify the risk events presented in Fig. 2. The goal was to find failure causes at the low level that put the system in risk condition and consequently could cause “insulin overdose” to the user.

We suggest organizing the fault tree at least in three decomposition levels: first level to identify the hazard to be analyzed; second level should be formed by labels that organize failure causes into groups; and third level to describe the failure causes.

Of course more levels can be created depends on the system complexity, but the levels between first and last are levels to organize failure causes into groups and not to describe them in details. As observed during this case study the failure causes were really identified at the last level in the fault tree.

Columns (1) and (2) presented in Table II were directly mapped from fault tree presented in Fig. 2. Column (1) corresponds to the second level of the fault tree and then the risk events identified by FTA were managed as high level safety requirements, which are: “incorrect input of insulin value”, “incorrect calculation of basal infusion frequency” and “failure in insulin delivery system”; column (2) corresponds to the third level of the fault tree and identify the failure causes that can lead the system to cause insulin overdose to the user.

TABLE II. SAFETY FUNCTIONAL REQUIREMENTS DERIVED FROM FTA.

| | | Safety Functional Requirements | |
|---|--|--|--|
| (1) Safety Requirements | (2) Failure causes | (3) “Should” Requirements | (4) “Should Not” Requirements |
| Incorrect input of insulin value | Failure in user interface (S) | 1. The system should delimit insulin value range during the specification of basal infusion profile. 2. The input of insulin value to basal infusion profile specification should be done using up and down buttons. | 1. The system should not allow the user to choose insulin value out of the safety specified range for basal infusion profile. |
| | Failure in data storing (S) | 3. The system should store at least 3 basal infusion profiles, each one specifying 24 insulin values (one by hour). | |
| Incorrect calculation of basal infusion frequency | Incorrect use of time interruption (S) | 4. The system should use the time interruption function offered by microcontroller to count time. | 2. The system should not continue basal infusion if “watch dog” timer is executed more than 5 times by hour. |
| | Failure in time interruption (S) | 5. The system should check if the time interruption is working as specified by user. | 3. The system should not continue basal infusion if the accumulated error between real time and specified time is more than 0.1%. |
| Failure in insulin delivery system | Motor driver failure (E) | 6. The system should implement acknowledge procedure to confirm the motor steppes performed. | 4. Motor driver should not failure during <i>bolus</i> infusion. |
| | Motor failure (M) | 7. The system should adopt a high precision motor to delivery insulin. | 5. Motor should not failure during <i>bolus</i> infusion. |

B. Classify Fault Tree Leaves as Software, Electronic or Mechanical Failure Causes

In safety-critical systems, particularly in embedded systems, usually there are three main types of failure causes: software, electronic and mechanical causes [15]. We consider that to improve the understanding of the safety requirements is very important to classify the failure causes according to these types.

Despite the FTA technique gives none suggestion about such classification we introduce this procedure as part of the adopted protocol because the recognition of the failure causes in a high level drives the investigation of the failures in a deeper way and consequently improves safety requirements specification. During the performed case study these

classification helped the requirements engineer to conduct the safety requirements elicitation with the right stakeholders.

C. Deriving Safety Functional Requirements in Style “Should” and “Should Not” Requirements

Safety functional requirements are those requirements that describe what actions and/or constraints should or should not be performed to maintain the system in a safe state. During the case study for each failure cause identified by FTA safety functional requirements were written in style “should” and “should not”. These requirements complemented FTA and offered more details to software engineers implement the software to control the insulin infusion pump.

Columns (3) and (4) in the Table II show “should” and “should not” requirements that were derived from the failure causes identified in the fault tree (Fig. 2). The failure causes guided the requirements engineer to conduct interviews with the stakeholders that participated in the study, i.e. the mechanical and electronic engineers and the doctor. Only failure causes described in fault tree do not offer sufficient information to derive safety functional requirements, but they proved to be a good base to begin the elicitation of the safety functional requirements with the stakeholders.

For instance, for the failure cause “failure in user interface” three safety functional requirements were described, of which two were “should” requirements and one was “should not”. During the elicitation of these requirements the doctor experience in diabetes treatment using insulin infusion pump helped requirements engineer to understand the necessity of basal infusion profile specification which defines valid insulin values to the continuous infusion process.

It is worthwhile to note that we tried to produce, as much as possible, requirements description using “should” and “should not” style. This can cause some redundancy but to the safety-critical system this kind of redundancy helps software engineers to better understand the safety requirements of interest to implement the control software.

D. Results from the Questionnaire Applied During the Study

With the intention to validate the performed study and also to offer evidences to answer the research questions defined in the beginning of this study – RQ1 and RQ2 - we prepared a questionnaire with five questions related to the usage of FTA to derive safety functional requirements. Two software engineers that participated in the study answered the following questions:

- **Q1:** Do you believe that the fault tree offers complete and sufficient information to guide you during the implementation of the control software of the insulin infusion pump?
- **Q2:** Do you feel comfortable adopting fault tree as the main source to understand the safety requirements to implement the control software of the insulin infusion pump?
- **Q3:** Do you consider that the safety functional requirements presented in style “should” and “should

not” are more adequate to drive you in the implementation of the control software of the insulin infusion pump than fault tree presented early?

- **Q4:** Do you think that “should” and “should not” requirements, being complementary in the safety functional requirements descriptions, improve the understanding of what should be implemented in the control software of the insulin infusion pump?
- **Q5:** As software engineer in charge to implement the control software of the insulin infusion pump, you prefer a software requirements document composed by:
 - Only fault tree
 - Only requirements in style “should” and “should not”
 - Both representations

To answer the question 1 to 4 the software engineers had to choose only one alternative available in style of Likert scale. The available alternatives were: totally agree, partially agree, partially disagree and totally disagree. The question 5 had three alternatives, as shown above (Q5). Table III presents the questionnaire results.

The software engineers that answered the questionnaire did not participate in the confection of the fault tree nor the “should” and “should not” requirements. They had a general view about the insulin infusion pump functionalities obtained from the literature. The FTA was done by the requirements engineer that participated in the study, which interacted with the electronic and mechanical engineers as well as the doctor to elicit the safety requirements.

TABLE III. ANSWERS FROM THE SOFTWARE ENGINEERS THAT PARTICIPATED IN THE CASE STUDY.

| Questions | Answers from the Software Engineer A | Answers from the Software Engineer B |
|-----------|--|--|
| Q1 | Partially agree | Partially agree |
| Q2 | Partially agree | Partially agree |
| Q3 | Totally agree | Totally agree |
| Q4 | Totally agree | Totally agree |
| Q5 | Only requirements described in style “should” and “should not” | Fault tree and requirements described in style “should” and “should not” |

The software engineers answered the questions Q1 and Q2 based on the fault tree presented in Fig. 2. Only after they answered these questions it was presented to them the requirements in style “should” and “should not”, as presented in Table II. They answered Q3 based on Table II, and finally they answered Q4 and Q5 considering the information available in Fig. 2 and Table II.

Although each software engineer answered the questionnaire alone, without communication between them, it is interesting to note that the answers to questions Q1 to Q4 were the same. Only the answers to Q5 were different, as can

be observed in Table III. The questionnaire result is analyzed as follows.

Both software engineers partially agreed that fault tree offered complete and sufficient information to guide him during the implementation of the control software (*Q1*). They also partially agreed to adopt fault tree as the main source to understand the safety requirements to implement the control software (*Q2*).

However, when the safety functional requirements presented in style “should” and “should not” were shown to the software engineers, both of them totally agreed that requirements in that way are more adequate to drive them to implement the control software than fault tree (*Q3*). The software engineers also totally agreed that requirements in style “should” and “should not” improve the understanding of what should be implemented in the control software (*Q4*), despite the redundancy that such style of description may bring.

Finally, when the software engineers were asked for preferences about the requirements document to guide them during control software implementation (*Q5*), none of them chose requirements document composed by only fault tree. Software engineer *A* chose requirements document composed by only requirements described in style “should” and “should not”, and software engineer *B* chose requirements document composed by both representations. These answers reveal that software engineers, in this case study, feel more comfortable using safety requirements described in style “should” and “should not” than FTA graphics representation.

E. Findings Emerged During the Application of the Adopted Protocol

In subsections *A*, *B* and *C* we reported the application of the proposed protocol to derive safety functional requirements to “insulin overdose” hazard. During the application of the protocol some findings emerged which are discussed as follows.

1) Finding: Information available in FTA is not enough to derive safety functional requirements.

This finding emerged during the study when the requirements engineer was trying to derive safety functional requirements from FTA. Despite the FTA to offer a good picture of the overall failure causes that can lead to analyzed hazard, when the requirements engineer was analyzing “insulin overdose” several important details to specify the safety requirements were missed. FTA offered a base to organize the safety requirements concerned to “insulin overdose”, but the details to mitigate the failure causes, both related to constraints and functionalities, were not available in the fault tree.

The details were obtained from the stakeholder that participated in the study. For instance, the details to derive safety functional requirements to mitigate the failure cause “*Incorrect use of time interruption*” were not available in the fault tree. But this failure cause guided the requirements engineer to elicit the necessary information with the electronic

engineer that participated in study, because he was the stakeholder that had the knowledge to derive the safety requirements to mitigate the failure cause under investigation. Thus, the requirement “*The system should use the time interruption function offered by microcontroller to count time*” – presented in Table II - only could be derived consulting the electronic engineer.

All the requirements presented in Table II, both in style “should” and “should not” were not directly available in the fault tree. To derive such requirements it was necessary to elicit them from the stakeholders that participated in the study. However, the failure causes identified using FTA were the drive to the elicitation process and helped the requirements engineer to derive the safety functional requirements to mitigate “insulin overdose”. This finding helped us to answer the RQ2. In fact in the performed study the FTA representation was not enough to derive safety functional requirements, albeit it had offered a base to guide the elicitation process that helped requirements engineer to derive the requirements. This base was formed by all failure causes identified by FTA.

2) Finding: The intermediate levels of fault tree help to organize safety requirements into groups.

Despite the intermediate levels of fault tree have been created to organize the FTA until the analyzers discover the failure causes at the last level – fault tree leaves – during the case study we noted that the intermediate levels also can help requirements engineers to organize the understanding about the analyzed hazard into groups of safety requirements.

The second level of the fault tree created during the case study – shown in Fig. 2 - helped the requirements engineer to organize the safety requirements concerned to “insulin overdose”. The safety requirements were organized into three groups, as shown in column (1) - Table II. The safety functional requirements were derived – columns (3) and (4) - following the organization defined by FTA, thus all “should” and “should not” requirements already had a group to be accommodated during the elicitation process.

3) Finding: The classification of the failure causes helps during the safety functional requirements elicitation.

During the study we include a new procedure when FTA was being performed. This procedure was to classify the failure causes as software, electronic or mechanical causes. The tags (S), (E) and (M) close to the failure causes in Fig. 2 show the classification done during FTA.

This classification helped requirements engineer during the elicitation process to derive safety functional requirements. For instance, the failure causes classified as software or electronic failures drove the requirements engineer to elicit safety requirements with the electronic engineer, which had experience both in software and electronic development. Mechanical failures drove the requirements engineer to elicit safety requirements with the mechanical engineer.

4) **Finding:** Requirements descriptions in style “should” and “should not” help software engineer to better understand the system safety requirements.

During the elicitation process we decided to derive the safety functional requirements in the style “should” and “should not” requirements. Such decision was based on two reasons: (1) it is a very common style to describe requirements and well known in the Requirements Engineering community; (2) we believe that “should” requirements could be complemented by “should not” requirements in many situations improving the general understanding about the safety requirements.

The answers from the software engineers that participated in the case study supported our assumption, as we can observe from the *Q4* results in the applied questionnaire. Even if some redundancy is present in the requirements it is accepted because the improvement in the understandability compensates the redundancy.

V. CONCLUSION

This paper presented results from a case study performed in a Brazilian company in the area of the electronic medical devices. During the case study a protocol to derive safety functional requirements was performed in the context of a prototype development of a low cost insulin infusion pump.

The findings that emerged during the case study and the answers from the software engineers that answered the questionnaire brought some evidences that helped us to answer the researches questions under investigation – RQ1 and RQ2.

Answering RQ1, we can say that based on the evidences from the performed case study FTA does not offer enough information to drive software engineers to implement safety requirements. According to the questionnaire results the software engineers did not feel safe to use only fault tree to understanding what safety requirements should be implemented in the control software. They considered that safety requirements described in style “should” or “should not” are more adequate than fault tree representation, or if fault tree is used it should come accompanied by safety functional requirements in style “should” or “should not”.

Answering RQ2, we can say that based on the evidences from the performed case study the fault tree resulting from the FTA is not a good enough representation to derive safety functional requirements, i.e. the information present in the fault tree is not sufficient to derive safety functional requirements. Fault tree representation proved to be very useful to organize safety requirements into groups, but it was necessary to appeal to the stakeholders to get necessary information to derive safety functional requirements, because only fault tree didn't supply the requirements engineer with necessary information to write safety functional requirements.

Despite the FTA is a well known technique used to analyze risks in safety-critical systems, we didn't find in the literature any formal protocol to derive safety functional requirements from FTA. We believe that such gap difficult requirements engineers to produce high quality safety requirements

specification when FTA is adopted. The proposed protocol adopted in the performed case study to derive safety functional requirements of course is not yet a formal protocol, but it can be improved to become one. Nevertheless, the proposed protocol in the performed case study helped the requirements engineer to derive safety functional requirements from FTA to the “insulin overdose” hazard.

In the context of researches in safety-critical systems we point out the following future work:

- To use the proposed protocol to derive safety functional requirements for other hazards in the context of the insulin infusion pump, which is under development in collaboration with a Brazilian company;
- To perform an evaluation of the adopted protocol inviting more requirements engineers to using of such protocol in other safety-critical systems;
- To create formal guidelines to help requirements engineers when using the adopted protocol to derive safety functional requirements from FTA;
- To replicate the performed case study in other safety-critical domains, e.g. in automotive and avionics domains.

REFERENCES

- [1] D. Firesmith, “Engineering Safety- and Security-Related Requirements for Software-Intensive Systems”, 32nd IEEE International Conference on Software Engineering, One-Day Tutorial (ICSE’2010). South Africa, May 2010.
- [2] P. Liggesmeyer and M. Trapp, “Trends in Embedded Software Engineering”, IEEE Software Magazine, v. 26, n. 3, 2009, pp. 19-25.
- [3] T. Sanislav, and L. Miclea, “Cyber-Physical Systems - Concept, Challenges and Research Areas”, Journal of Control Engineering and Applied Informatics (CEAI), vol. 14, 2012, pp. 28-33.
- [4] I. Sommerville, Software Engineering. Addison-Wesley, 9th edition, 2010.
- [5] B. S. Medikonda and S. R. Panchumarthy, “A Framework for Software Safety in Safety-Critical Systems”, SIGSOFT Software Engineering Notes, vol. 34, n. 2. March 2009. pp. 1-9.
- [6] D. Firesmith, “Engineering Safety Requirements, Safety Constraints, and Safety-Critical Requirements”, Journal of Object Technology, vol. 3, n. 3, march/april 2004.
- [7] Y. Zhang, P. L. Jones and R. A. Jetley, “Hazard Analysis for a Generic Insulin Infusion Pump”, Journal of Diabetes Science and Technology, vol. 4, n. 2, March/2010. pp. 263-283.
- [8] S. Yaturu, “Insulin Therapies: Current and future trends at dawn”, World Journal of Diabetes, vol. 4, n. 1, February/2013. pp. 1-7.
- [9] S. Kumari, G. Kondeti, S. Pakki, T. L. V. Chandrasekhar and S. Balu, “Method of Safety Critical Requirements Flow in Product Life Cycle Processes”, International Conference on Integrated Communications, Navigation and Surveillance Conference (ICNS), May/2011. pp. N2-1 – N2-4.
- [10] D. Gollub, M. Kretschmer and A. Heyl, “Defining Safety Requirements for Hybrid Electric Vehicles Using System Simulation”, 3rd IET International Conference on System Safety, Oct/2008. pp. 1-5.

- [11] S. B. Driskell, J. Murphy, J. B. Michael and Man-Tak Shing “Independent validation of software safety requirements for systems of systems”, 5th International Conference on System of Systems Engineering (SoSE), June/2010. pp. 1-6.
- [12] M. J. Squair, “Issues in the application of Software Safety Standards”, 10th Australian Workshop on Safety Related Programmable Systems (SCS), vol. 55. Sidney, 2005. pp. 13-26.
- [13] E. Sikora, B. Tenbergen and K. Pohl, “Industry Needs and Research Directions in Requirements Engineering for Embedded Systems”, Requirements Engineering Journal, vol. 17, n. 1, 2012. pp. 57-78.
- [14] E. Navarro, P. Sánchez, P. Letelier, J. A. Pastor and I. A. Ramos, “Goal-Oriented Approach for Safety Requirements Specification”, Proceedings of the 13th IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS), 2006. pp. 318-326.
- [15] F. Vahid, T. Givargis, *Embedded System Design: A Unified Hardware/Software Design*. John Wiley & Sons, 2002.
- [16] R. R. Lutz, A. Patterson-Hine, C. R. Frost, S. Nelson, D. Tal and R. Harris, “Using Obstacle Analysis to Identify Contingency Requirements on an Unpiloted Aerial Vehicle”, Requirements Engineering Journal, vol. 12, n. 1, 2007. pp. 41-54.
- [17] F. Modugno, N. G. Leveson, J. D. Reese, K. Partridge and S. D. Sandys, “Integrated Safety Analysis of Requirements Specifications”, Requirements Engineering Journal, vol. 2, n.1, 1997. pp. 65-78.
- [18] E. J. Broomfield and P. W. H. Chung, “Safety Assessment and the Software Requirements Specification”, Journal of Reliability Engineering and System Safety, Elsevier. Vol. 55, n. 3, 1997. pp. 295-309.
- [19] R. R. Lutz and I. C. Mikulski, “Operational Anomalies as a Cause of Safety-Critical Requirements Evolution”, Journal of Systems and Software, Elsevier, vol. 65, n. 2, 2003. pp. 155-161.
- [20] R. R. Lutz and I. C. Mikulski, “Empirical Analysis of Safety-Critical Anomalies During Operations”, IEEE Transactions on Software Engineering, vol. 30, n. 3, 2004. pp. 172-180.

The DODT Tool Applied to Sub-Sea Software

Tor Stålhane

Dept. of Computer and Information Science
Norwegian University of Science and technology
Trondheim, Norway
stalhane@idi.ntnu.no

Tormod Wien

Corporate Research
ABB
Oslo, Norway
tormod.wien@no.abb.co

Abstract—Using natural language is still a common form of writing software requirements. Tools and techniques to improve the quality of natural language requirements may give better results than attempts to convince industry to use something else. We have combined natural language requirements with tool support using boilerplates and domain ontologies, enabling detection of ambiguities and incompleteness in requirements. This paper reports on a case study where requirement analysts used the developed tool to analyse requirements for a safety-critical control system. The experience showed that people were able to use the tool to develop a domain ontology and apply boilerplates to describe requirements in a structured way, yielding requirements readable for humans and analysable for the tool. The tool support improved the quality of requirements by reducing ambiguities and inconsistent use of terminology, removing redundant requirements, and improving partial and unclear requirements.

Index Terms—requirements specification, natural language, patterns, ontologies

I. INTRODUCTION

When developing a software system, the requirements are the basis for all later work. If the requirements are wrong, imprecise or incomplete, the same will hold for the final system. On the other hand, if the requirements are complete, consistent and unambiguous, the final system will usually have a high quality. More than 40% of all problems in a typical software project may be related to requirements [1]. The main reason for this is that the developers have nothing, except the users and applicable standards to check the requirements against. Later in the process it is possible to compare the design against the requirements, the code against the design and so on, to make sure that nothing that is needed is lost and nothing that is not needed is added.

Since the requirements are an important communication medium, they need to be understood by all stakeholders. Thus, we have chosen natural language (NL) as a means of specifying requirements. If we want to use tools to formally check the requirements' quality, we will need a formal NL presentation. We have selected the boilerplates representation to achieve this.

The rest of this paper is organized as follows: Section II gives a short background while section III discusses related work. Section IV gives an overview of the research methodology used and presents the research the questions while Section V shows our analysis of ABB's experiences with the tool and thus the answers to the three research

questions. The paper is rounded off with Section VI which presents our conclusions and Section VII where we discuss where to go from here.

II. BACKGROUND

A. The DODT Tool

The methods and tool discussed in this paper all stem from the CESAR project – Cost-efficient methods and processes for safety relevant embedded systems – which is a European funded project from ARTEMIS Joint Undertaking (JU). The project had requirements handling as one of its top priority items. To achieve this, the project developed the DODT (Domain Ontology Design Tool) for requirements analysis. This tool is a further development of the GNLQ (Guided Natural Language) tool which is now available via SourceForge. Even though this approach helped requirement engineers to develop better requirements, we were not able to attack important quality issues.

One of main goals for the new DODT tool is the ability to check requirements for completeness, consistency and unambiguity, based on information in domain ontologies. For the DODT tool, completeness means “covers all relevant elements in the system ontology used”.

We need two concepts to describe the DODT – ontologies, and boilerplates [2]. They interact with the tool as shown in Fig 1. One of the first to use boilerplates for software requirements were Hull and Dick [3]. The main reason for introducing boilerplates was to add structure to the NL requirements. A general requirement formulated using a boilerplate has three parts [3] – a precondition, an active part and an action-condition. We use the “<” and “>” to isolate the variable parts of the boilerplate requirement.

- Pre-condition – one or more operational conditions that need to be true in order to move on to the action part of the requirement, e.g. “While <state>...” or “If <event>...”
- Active part - one or more actions performed or abilities made available if the pre-condition is true, e.g. “...<system> shall <action>” or “...<system> shall allow <entity> to be <state>”
- Action-condition – one or more operational conditions or restrictions that apply to the action or the way the action is performed, e.g. “...without <action>” or “...within <number> <unit>”

For a boilerplate, an operational condition is either the occurrence of an event or that the system is in a defined state. Using boilerplates we get a set of semi-formal requirements. These requirements are both easy to read and possible to handle by software tools. An example of a complete boilerplate requirement can be written as shown below.

If <event>, <system> shall <action> within <number> <unit>.

This template can be used to write the following requirement: **If <temperature reaches max temperature>, <boiler controller> shall <reduce power to heating unit> within <2> <seconds>.**

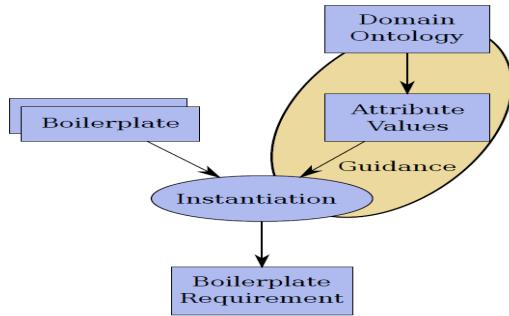


Fig. 1 DODT overview

Domain ontologies make the tool more than just a database for requirements [4]. We can make sure that similar requirements are written using the same structure and that each term is interpreted correctly. If we go back to the example above, there is no doubt as to how <2> <seconds> shall be interpreted since it is preceded by the term **within**.

Ontologies have three parts – a thesaurus, a set of relationships between terms in the thesaurus and rules for making inferences based on these relationships. The information needed to build ontologies is in our case mainly found in relevant standards and glossaries. Building ontologies are at the present not an automatic process. Fig. 2 shows part of a steam boiler controller ontology

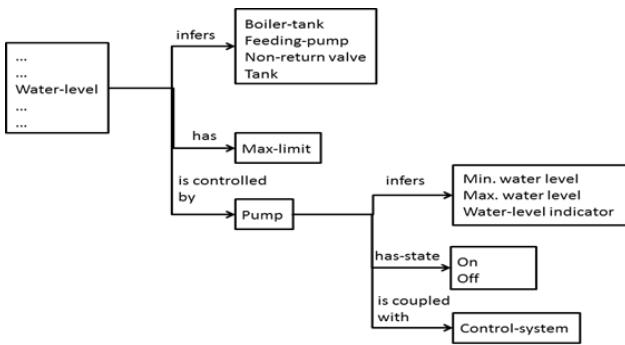


Fig. 2 Part of a steam-boiler ontology

The DODT uses ontologies for three purposes:

- To check requirement quality.

- To handle synonyms. E.g. in a steam-boiler ontology, the terms tank, vessel and boiler-tank will be synonyms.
- For safety analysis. Each ontology entity linked to a physical unit or a system component has a set of generic failure modes.

In the DODT tool, requirement quality is defined by the following metrics:

- Completeness: Check for missing requirements. The analysis reports all terms that are linked via relations from requirement terms and that do not occur among the requirements
- Inconsistency: Terms occurring in the requirements and linked by a contradicting axiom or requirements containing identical sets of terms and exactly one cardinal which is different in the two requirements.
- Ambiguity: Using terms that have multiple interpretations despite the reader's knowledge of the context. It does not matter whether the author unintentionally introduced the ambiguity, but knows what was meant, or he or she intentionally introduced the ambiguity to include all possible interpretations
- Noise: Terms not contained in the domain ontology
- Opacity: Unrelated terms in a requirement which possibly means that a concept is used erroneously
- Similarity: Requirement pairs that is similar by sharing a configurable number of concepts. Alternatively the analysis is also able to detect requirement pairs that use exactly the same set of terms
- Obsoleteness: Requirements using obsolete ontology concepts

The DODT tool handles ambiguities by looking for a term with the same meaning which is already in the ontology. If another term is defined in the ontology to precisely represent the intended concept the term from the ontology should be used instead. If the less precise term is defined as a possible synonym to the preferred term, the tool could be able to propose this change to the analyst.

The DODT tool handles completeness by ensuring that all ontology components are referred to in one or more requirements. If we consider the ontology shown in Fig. 2, we see that if we have a requirement which mention the term "water level", the tool will require that we also have requirements that refer to the boiler-tank, a feeding pump and a non-return valve. Otherwise, the tool will indicate that you have one or more missing requirements.

B. Case Study Context

ABB has already decided to use the DODT tool to improve requirements handling, especially to reduce ambiguity, improve communication and understanding and avoid unclear and redundant requirements. ABB therefore wanted to assess the tool's supportiveness when it comes to removing ambiguous and redundant requirements, different term used to describe the same "thing", unclear or ambiguous terms and

partial or unclear requirements. ABB's needs are covered by the part of the DODT tool's problem identification areas "inconsistency" and "ambiguity".

III. RELATED WORK

A. Use of NL in Requirements

There has been a lot of work done on the use of NL in requirements specification – see the work of Fabbrini et al. [5] which sums up some of the most important contributions. It is possible to use unconstrained natural language and then apply a tool to check for inconsistencies, ambiguities and so on. A case study performed by Fabbrini et al. shows promising results in that the tool QuARS was able to identify several common problems such as vague and underspecified requirements [6]. For the 444 requirements for a space application written in NL they found 257 problems, and among these there were 17 vague requirements and 48 requirements that were underspecified.

The EARS templates (Easy Approach to Requirements Syntax), which has been developed by Rolls Royce [7], and the later version BIG EARS [22] have many ideas in common with boilerplates. Just as with boilerplates, EARS allows the user to combine templates to create more complex requirements than can be done with each of the templates alone. However, they warn people that the strong side of EARS – its simplicity – easily can be compromised by creating too complex requirements. As an alternative, they recommend using truth tables when the requirement pre-conditions get overly complex. A case study on the use of EARS, reported in [7] shows that compared to unstructured text, EARS give requirements that are easier to test and use as a basis for implementation, has less duplicates, omissions and complexity and the requirements are less ambiguous and vague. EARS has just two weaknesses: (1) it has no tool support and (2) since it does not contain domain knowledge, it is not able to check for requirement quality factors such as completeness, consistency and ambiguity.

SPS is a pattern-based, restricted English grammar [8]. It is structured as a set of logic propositions rather than as NL statements. For instance: "Globally it is always the case that if P holds, then S holds after at most c time units". Here P and S are Boolean proposition values and c is an integer. The SPS is assessed through a case study by A. Post et al. [9]. The case study showed that only 25 out of 250 requirements used in the case could *not* be expressed using SPS.

B. Use of Ontologies in Requirements

One of the languages that have been used for requirements is PSL – Process Specification Language – see Gruinger and Menzel [10]. The tool uses ontologies but only to define the requirements structure. No domain ontology is involved. Bloem et al. have constructed the tool RATSY using this language [11]. The tool can help software engineers to develop a property based design based on the PSL specification. The weak side of this approach, beside the lack of a domain ontology, is that the resulting specification is just as complex

as the code, there is no NL support and no domain knowledge included.

Siegemund et al. [12] give an overview of the challenges and opportunities when using ontologies in requirements engineering. The authors focus on how an ontology can help the requirements engineer to identify e.g., consistency and completeness. The ODRE (Ontology-Driven Requirement Engineering) meta-model for requirements, presented in [12] only contains a requirement ontology, not a domain ontology. Falbo et al. [13] has also introduced a requirement ontology, based on the unified foundational ontology – see [14]. Again, this is a requirements ontology and not a domain ontology. The same is true for Innab et al.'s work [15]. Thus, these methods are of limited use to the working requirement engineer.

Two papers – one by Castaneda et al. [16], and one by Kaiya and Saeki [17] – both apply domain ontologies and requirements ontologies in requirements engineering. None of them have developed a tool but are able to demonstrate that their approaches will help to identify requirements problems such as incompleteness and inconsistencies.

No tool or approach that we have identified (1) allows NL requirements, (2) has a set of templates to structure the NL requirements and to enable requirements analysis and (3) has a **domain** ontology which can be used to analyse the requirements' quality. The DODT, however, meets all of these needs.

IV. RESEARCH METHODOLOGY

A. Research Questions

To see if the DODT can fulfil the needs identified in section III and to evaluate the DODT's usefulness to ABB, we need answers to three research questions:

- RQ1: Can we move requirements from Excel or Word to DODT without losing important information? See section V.A.
- RQ2: Can we use DODT to build a useful ontology based on free text requirements and applicable standards? See sections V.B and V.C
- RQ3: Will the DODT help the engineers to develop a requirement set that is consistent and unambiguous? See section V.D

B. Research Design and Operation

For the evaluation ABB selected the requirements specification originating from the ISO 13628-6 [18] standard for subsea installations. These requirements were reformulated and new requirements were added to take into consideration customer requirements and ABB internal requirements and constraints. This resulted in 97 system specific requirements, which were the starting point for our evaluation process. In order to answer the three research questions we performed the relevant project activities and collected qualitative and quantitative data.

The requirements were originally on a tabular format comprising identifier, headline, requirement text, motivation for the requirement, the source of the requirement, e.g. internal

(ABB) or a standard, customer and priority. The requirements were converted to boilerplates and stored into the DODT tool as boilerplate requirements – see Fig. 1.

In order to transform the requirements in the DODT tool from a natural language presentation to boilerplates the RE used the process shown below. Note the tight coupling between the requirements insertion process and the ontology development process.

- Build the ontology based on the NL requirements and the relevant standards
- Step through the requirements one by one, rewriting when necessary, and adding concepts to the ontology when missing concepts are encountered. The tool will give assistance here but the selection of the appropriate boilerplate is up to the requirements engineer.
- Use the requirements analysis feature of the tool to search for incomplete, unclear or ambiguous requirements
- Create a list of unclear concepts and/or requirements that couldn't be resolved on the fly. This list is sent to the subsea systems team for clarification.
- Transform selected requirements into boilerplates

As mentioned earlier, ABB's main goals were to identify inconsistencies and ambiguities in the requirements. This was done using the DODT's requirements analysis capabilities.

C. Validity Evaluation

In general we need to consider three groups of validity threats: Threats to conclusion validity, threats to construct validity and threats to external validity. Since this is a case study and not an experiment, issues related to construct validity are not present.

Internal validity – did the observed effect really stem from the treatment, in this case, from the use of the DODT? Since the observed positive effects are related to the ontology and to a partly automation of requirements handling – see section VI – we are confident that the observed effects stem from the use of the DODT.

Conclusion validity is concerned with data quality. The qualitative data collected during this case study were the RE's experiences of using the tool. The quantitative data were collected from documents generated during the requirements process. In both cases the data are representative for the engineer's use of the DODT on the requirements for the system under consideration.

External validity is concerned with generalizations of the results, both within the company and to other companies in the same application domain. Development of software for a subsea installation is no different from the development of other safety-critical software considering IEC 61508 or other, similar standards – e.g. ISO 26262. We thus see no threats to the conclusion validity.

V. RESULTS AND ANALYSIS

A. Requirements from Word and Excel to DODT

Before using DODT, all requirements were written in Excel spread sheets or in Word documents. A typical example from the Subsea Requirements Specification is shown in Table 1. ABB wanted to preserve this information when importing the requirement into DODT. Since the DODT requirement format supports all fields in the table above – see Fig. 3 below – it was a simple task to map the requirements from Table 1 to similar fields in the DODT requirements format. The requirements in the Word document were imported into Excel. We then used an Excel macro to format each requirement according to the column headers as shown above and then import the Excel tables into DODT. The final result of this process was 97 requirements in the DODT tool.

Table 1 ABB example requirement format

| ID | Headline | Pri |
|---|----------------------------|-----|
| DLV-01.03.01 | SUBSEA FACTORY SYSTEM SIZE | 1 |
| Requirement | | |
| The ABB subsea control system shall be configurable to control and monitor a subsea factory consisting of : | | |
| <ul style="list-style-type: none"> - Multiple well templates - Multiple separation and boosting processes - Subsea power electronics for distribution and conversion | | |
| Motivation | | |
| Typical contents of an installation according to Market Case 3. | | |
| NOTE: It is assumed that a subsea factory is split into several template clusters separated up to 50km, and that each template cluster require several control nodes. | | |
| Source ABB | | |
| ABB | | |

RQ1 can thus be answered in the affirmative – we can move requirements from Word documents and Excel spread sheets to DODT without losing information.

B. Converting Requirements to Boilerplate Representation

The basis for the DODT [19] is the boilerplates. Boilerplates as a means for software requirements specification were introduced by Hull and Dick [3] and are a set of requirements templates. The main reason for introducing boilerplates was to add structure to the NL requirements. In this way, there is a limited set of ways to formulate a requirement which again helps us to ensure a consistent representation. The CESAR project has developed this concept

further and the current set of requirement boilerplates has 34 items and is used by e.g., Infineon, Airbus (G) and ABB – CR.

Fig. 3 ABB subsea requirements imported into DODT

When we started this work, no ontology existed for the subsea equipment domain. As we have observed in the CESAR project [20] it is difficult to build a complete ontology from scratch. The ontology was thus built step by step – starting with the abbreviations and definitions used in the originating requirements specification, based in ISO 13628-6 [18] and on concepts used in each requirement – adding new concepts, relations and axioms as the needs arose. The ontology was developed using the process described in section IV B.

We started the process described in IV.B with the 40 most used concepts. During the process, four concepts were removed, six concepts were changed and 18 new concepts were added. The final ontology contained 54 concepts.

When a requirement is entered into the DODT tool, all nouns are marked as green (found in the ontology) or red (not found in the ontology). This information eases the work of defining and adding new concepts to the ontology.

The task of creating an ontology must carefully consider every proposed concept to decide whether it belongs to the specific domain. As an example, consider the following requirement:

"The SCPS shall have an architecture and system design which enables maintenance and replacement of parts of the system without affecting the rest of the installation."

In the example the term SCPS (Subsea Control and Protection System) will be part of a “subsea ontology” while the other terms are generic.

The answer to RQ2 – Can we build a useful ontology based on free text requirements and applicable standards – has two parts:

- It is possible to build the main structure of an ontology based on the standards applicable to the domain in question.
- This ontology must be extended and refined as we insert new requirements into the DODT tool.

Fig.4 Ontology screenshot

Thus, we can use the standards to give us a good starting point but building a complete ontology for a domain is an iterative process.

C. Requirements Analysis

The requirements analysis feature in the tool is important to ABB. Fig. 5 shows the metrics view for the first version of the requirements – see Section II for definitions. We see that the tool found 96% noisy terms in the original set of requirements, i.e., terms that are not defined as concepts in the ontology. It also indicates that there were 15% ambiguous requirements and that 69% of the requirements have unrelated concept pairs used within one requirement (Opacity). The individual tabs provide a mean to go into each requirement in details and to decide if a new concept should be added to the ontology, if the term should be changed to a concept already defined, i.e. rewrite the requirement, or if it should just be ignored.

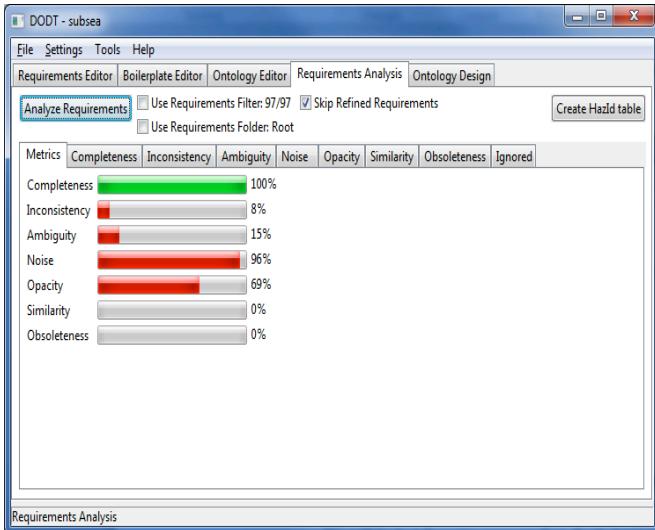


Fig. 5 DODT Metrics screenshot

The requirement engineer studied the noisy concepts first, firstly because they in most cases are simple to handle and secondly because they are important for the ontology enhancement. The DODT can show all noisy concepts – see Fig. 6. Some of the terms are obvious noise terms – e.g., ABB and note – and should be removed. Other terms are needed and must be inserted into the ontology – e.g., canister and Ethernet.

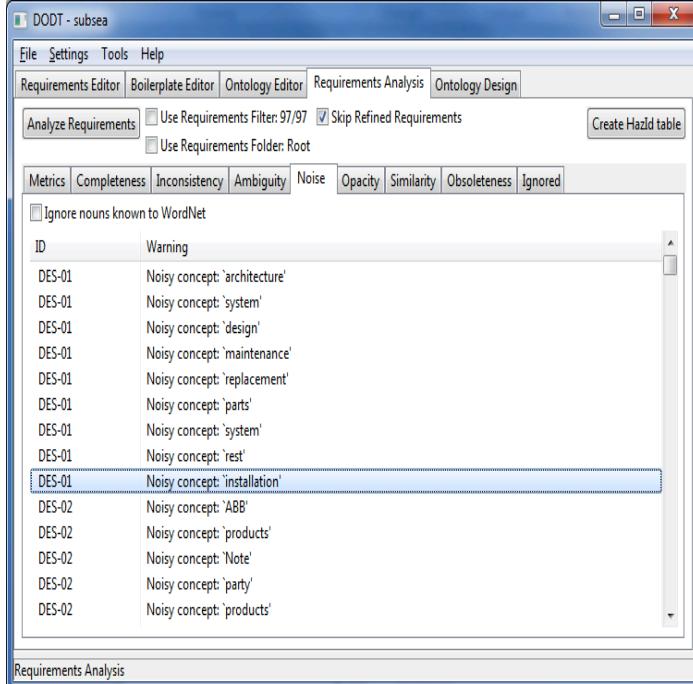


Fig. 6 Noisy concepts

D. Using the Ontology – Ambiguity and Opacity

1) *Ambiguity*: As can be seen in Fig 5, 15% of the requirements are ambiguous. The screen shot in Fig. 7 shows the ambiguous requirements – for instance requirements with

the ambiguous concept communication link. The reason for this being ambiguous is that the requirements engineer has defined specialized subclasses of this concept. It is now possible to look into the requirements where this concept is used in order to search for a resolution. From the screenshot in Fig. 7 we see that the concept communication link is used in the requirement DES-06:

“There shall exist a secondary communication link between the SCPS and the topside facility, either over single-mode optical fibre or on the power lines”.

This requirement refers to a communication link that can be realized using either optical fibres or power-line modems. In this case the requirements engineer therefore selects to keep the generic concept. In the tool this can be done by marking the requirement as "ignored" in the ambiguity tab, meaning that it will not be taken into consideration in the subsequent analysis. The next communication link concept is in requirement DLV-01.01.01:

“The SCPU shall include...Redundant communication link over Ethernet TCP/IP to topside and/or subsea control system(s)”

The requirement requires a redundant communication link, but it doesn't say anything about the physical media. Is it fibre optic, power-line modem or maybe both? What is meant by the term “redundant”? This requirement goes into the list of terms that need subsea system group clarification.

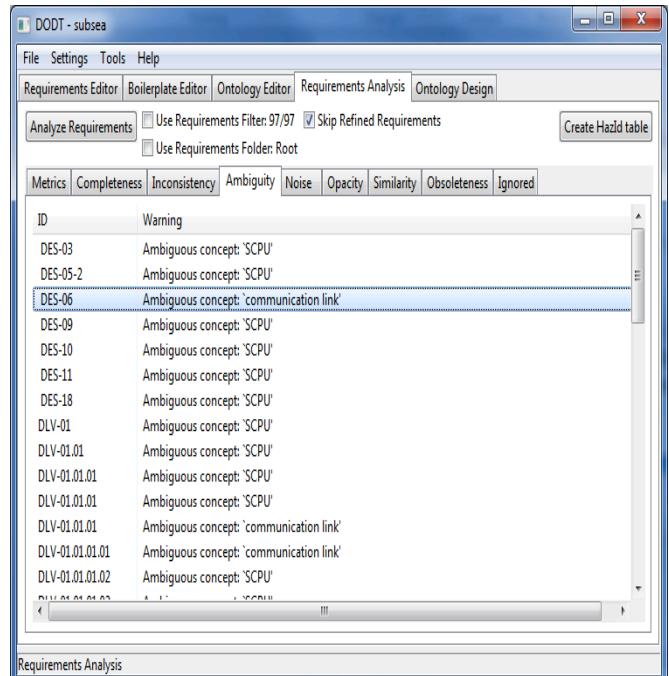


Fig. 7 Ambiguous requirements

We can go through all items in the list of requirements comprising ambiguous concepts. The requirements engineer is

offered three options: (1) edit the requirement, (2) ignore it, i.e., keep as is, or (3) replace it with one of the subclass concepts. The ambiguity view offers an efficient way of going through requirements and either keeping it in places where appropriate or replacing the concept with a more precise subclass concept.

2) *Opacity*: The first iteration with just a few concepts, also found some concepts marked as unrelated, meaning that the concept may be used erroneously in the requirement. Fig. 8 shows the tool's screen for opacity. Fig. 5 shows that at the start of the process there is 66% opacity – e.g., SCPU together with template and SCPU together with SCPS. One of the requirements contains the two unrelated concepts template and subsea interface unit:

“Templates shall be interconnected (via subsea interface units) with redundant, bidirectional fibre-optics, and communicate over Ethernet TCP/IP”.

The question is if there is a relation between the two concepts. The definition of *template* shows that this is *a large steel structure which is used as a base for subsea structures such as...etc*. That is, a *template* is a structure for holding all physical entities related to a subsea installation.

It is possible to extend the *template* definition with the axiom *contain*, so that the *template <contain> subsea interface units* (and other units). Fig. 9 shows the new ontology, where the concept tree is extended. The concepts *SCPU* and *subsea interface unit* are now *contained* in the template.

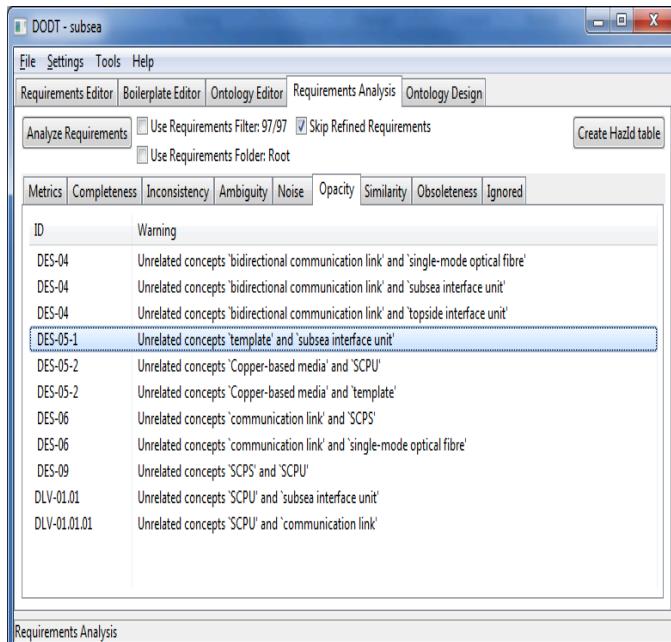


Fig. 8 DODT unrelated concepts

If the requirements analysis is run again, those requirements disappear from the opacity table. The DODT requirements analysis feature helps us to identify opacities and

enables us to relate new concepts, thus further contributed to clearly and precisely written requirements.

Of the 97 requirements that we started with, 35 requirements have undergone minor changes, e.g. changed one or more terms in order to comply with the ontology. Ten requirements have been completely rewritten for the same reason. In the end, this left us with 90 requirements. Thus 50% of all requirements were changed during the requirements analysis process.

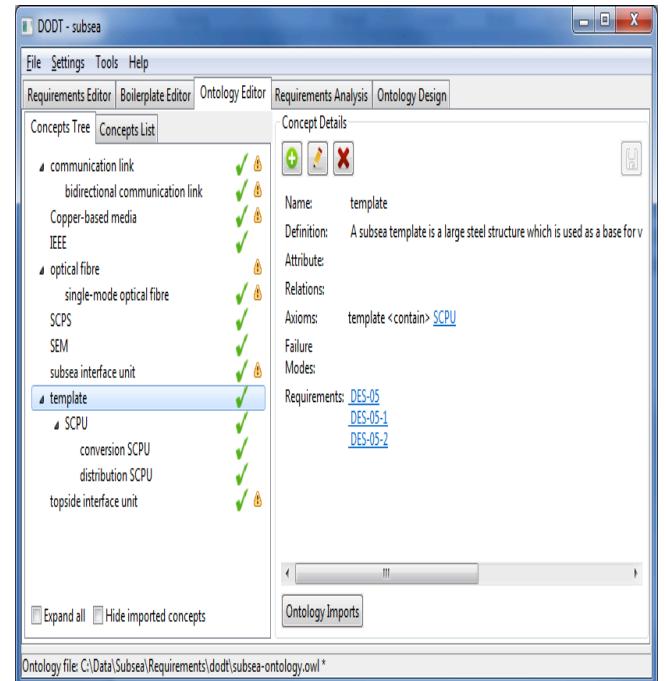


Fig. 9 DODT concept definition example

We will conclude that the DODT tool help the requirements engineers to identify ambiguities and opacities. Thus, RQ3 is answered in the affirmative.

VI. CONCLUSIONS

The final result of the requirements conversion process was a complete set of requirements in DODT. Thus, RQ1 can be answered in the affirmative – we can move requirements from Word and Excel to DODT without losing important information. When it comes to R2, the answer is that yes, we can develop an ontology based on applicable standard and free text requirements but only in an iterative way – start with an ontology based on applicable standards, add requirements, adjust the ontology, add more requirements and so on.

The answer to RQ3: “Will the DODT tool help the engineers to develop a requirement set that is consistent and unambiguous” is definitively “Yes”.

We have also seen that the tool has the potential to improve communications – in our case between requirements engineers and domain specialists. See also the discussion at the end of V.C

The qualitative information collected indicates that the DODT helps developers to work in a more systematic way, formulate requirements more concisely, build a domain ontology and identify and correct requirement opacities and ambiguities. The quantitative information shows that building an ontology is an iterative process and that a considerable part – in our case 50% – of the original requirements may need to be re-written in the process.

Our experience is in line with that of other companies – see e.g. the experiences reported by Armagaud from AVL [21].

VII. FURTHER WORK

The work on boilerplates and ontologies will be tested out in more case studies. In order to make any progress we need companies which have data that can be used as a baseline. We are especially interested in doing future research on the following topics:

- The number of inconsistent or opaque requirements that reaches the implementation stage.
- How much of the DODT effect stems from the ontology and how much which stems from the use of templates. We already know that Rolls Royce has had good experience with EARS, which is just a set of templates, while ABB claimed that most their benefits from the DODT stems from the requirements analysis based on the domain ontology.
- Will the use of DODT improve the communication and cooperation between the requirements engineers and the domain specialists

We have started to look for appropriate cases for such studies.

REFERENCES

- [1] Standish Group: The Standish Group Chaos Report 2003
- [2] S. Farfeleder, T. Moser, A. Krall, T. Stålhane, H. Zojer, Ch. Panis: DODT: Increasing Requirements Formalism using Domain Ontologies for Improved Embedded Systems Development. Vortrag: 14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2011), Cottbus, Germany
- [3] Hull, E., K. Jackson, and J. Dick, Requirements Engineering. 2010, London: Springer Verlag
- [4] Farfeleder, S., Moser, T., Andreas Krall A., Stålhane, T., Omoronyia, I., Zojer, H.: Ontology-Driven Guidance for Requirements Elicitation
- [5] Fabbrini, F., Fusani, M., Gnesi, S. and Lami, G.: The Linguistic Approach to the Natural Language Requirements Quality: Benefit of the use of an Automatic Tool. Proceedings of the 26th Annual NASA Goddard Software Engineering Workshop
- [6] Fabbrini, F., Fusani, M., Gnesi, S. and Lami, G.: Quality Evaluation of Software Requirements Specification. Proceedings of Software & Internet Quality Week 2000 Conference, San Francisco, USA
- [7] Mavin, A., Wilkinson, P., Harwood, A. and Novak, M.: EARS (Easy Approach to Requirements Syntax). Proceedings of the 17th International requirements Engineering Conference
- [8] Konrad, S. and Cheng, B.H.C.: Real-time Specification Pattern. Proceedings of the 27th international conference on Software Engineering
- [9] Post, A., Menzel, I., and Podelski, A.: Applying Restricted Grammar on Automotive Requirements – Does It Work? A Case Study. REFSQ'11 Proceedings of the 17th international working conference on Requirements engineering: foundation for software quality
- [10] Michael Grüninger, Christopher Menzel: The Process Specification Language (PSL) Theory and Applications. AI Magazine 24(3): 63-74 (2003)
- [11] R. Bloem, A. Cimatti, K. Greimel, G. Hofferek, R. Könighofer, M: Roveri, V: Schuppan, and R: Seeger: RATSY - A New Requirements Analysis Tool with Synthesis. CAV 2010: 425-429
- [12] Siegemund K., Thomas, E.J., Zhao, Y., and Assmann U.: Towards Ontology-driven Requirements Engineering. Proceedings of 7th International Workshop on Semantic Web Enabled Software Engineering (SWESE), October 2011.
- [13] Falbo, R.A. and Nardi, J.C.: Evolving a Software Requirements Ontology. Em: XXXIV Conferencia Latino Americana de Informática - CLEI'2008
- [14] Guizzardia, G., and Wagner, G.: A Unified Foundational Ontology and some Applications of it in Business Modeling. CAiSE Workshops (3) 2004
- [15] Innab, N., Kayed, A., and Sajeev, A.S.M.: An Ontology for Software Requirements Modelling. 2012 IEEE International Conference on Information Science and Technology, Wuhan, Hubei, China; March 23 – 25, 2012
- [16] Castaneda, V., et al.: The use of Ontologies in Requirements Engineering, Global Journal of Research in Engineering, vol. 10, issue 6, November 2010
- [17] Kaiya, H. and Saeki, M.: Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. In Proc. of the 5th International Conference on Quality Software (QSIC), pp. 223-230, Melbourne, Australia. Sep. 19-21. 2005.
- [18] ISO: Petroleum and natural gas industries – Design and operation of subsea production systems. ISO 13628-6:2002
- [19] S. Farfeleder, T. Moser, A. Krall, T. Stålhane, H. Zojer, Ch. Panis: DODT: Increasing Requirements Formalism using Domain Ontologies for Improved Embedded Systems Development. Vortrag: 14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2011), Cottbus, Germany
- [20] Rajan, A., Wahl, T.: CESAR - Cost-efficient Methods and Processes for Safety-relevant Embedded Systems Springer, 25. March 2013
- [21] Armnagaud, E. et al.: Integrated tool-chain for improving traceability during the development of automotive systems. Proceedings of ERTS² 2012
- [22] Mavin, A. and Wilkinson, P.: “BIG EARS (The Return of “Easy Approach to Requirements Syntax”). 2010, 18th IEEE International Requirements Engineering Conference.

Towards Feature-Oriented Requirements Validation for Automotive Systems

Jiale Zhou¹, Yue Lu¹, Kristina Lundqvist¹, Henrik Lönn², Daniel Karlsson², Bo Liwång³

¹School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden

²Advanced Technology and Research, Volvo Group, Göteborg, Sweden

³Swedish Radiation Safety Authority (SSM), Stockholm, Sweden

¹{zhou.jiale, yue.lu, kristina.lundqvist}@mdh.se

²{henrik.lonn, daniel.b.karlsson}@volvo.com

³bo.liwang@ssm.se

Abstract—In the modern automotive industry, feature models have been widely used as a domain-specific requirements model, which can capture commonality and variability of a software product line through a set of features. Product variants can thus be configured by selecting different sets of features from the feature model. For feature-oriented requirements validation, the variability of feature sets often makes the hidden flaws such as behavioral inconsistencies of features, hardly to avoid. In this paper, we present an approach to feature-oriented requirements validation for automotive systems w.r.t. both functional behaviors and non-functional properties. Our approach first starts with the behavioral specification of features and the associated requirements by following a restricted use case modeling approach, and then formalizes such specifications by using a formal yet literate language for analysis. We demonstrate the applicability of our approach through an industrial application of a Vehicle Locking-Unlocking system.

Index Terms—feature-oriented requirements modeling; model-based requirements validation; eTASM; RUCM; software product lines; systems functional behaviors and non-functional properties

I. INTRODUCTION

With the growing maturity and standardization of the automotive domain, requirements specifications for automotive systems tend to center around the concept of feature models. Feature models [1] are proposed to capture the commonality and variability within a software product line by using features, between which there are relations and constraints. Further, a feature [2], [3] is a logical unit of functionality comprehensible to end-users, which consists of the requirements (i.e., the feature requirement hereafter) associated with the feature and the corresponding behavioral specification (i.e., the feature behaviors hereafter). A product can be configured by selecting a set of features (i.e., the feature set hereafter) from a feature model. The validity of a feature set refers to two situations: 1) one is from the structural perspective, i.e., the selected features should conform to the constraints defined by the feature model and, 2) the other is from the functional perspective, i.e., no undesirable behaviors exist between two or more (as integrated) feature behavioral specifications. In order to increase the confidence of the validity of the feature set, several feature-oriented requirements validation techniques [3], [4], [5], [6], [7], [8] have been developed. However, it is well recognized that with

the increasing size of feature models, the inherent variability of the feature sets leads to an inevitable issue that the hidden flaws of features are difficult to avoid [9]. Especially, the feature interaction problem (referring to the situation that two or more features exhibit unexpected behaviors) cannot be detected when the features are used in isolation.

As the unexpected behaviors can result in uncertainties and even hazards of automotive systems, adequate efforts on detecting the unexpected feature interactions thereby must be applied in the early stages of the pertaining development process. In the literature, there are many examples [7], [10], [11] where the process starts with translating the natural language specification (NLS) of a feature into a formal language specification (FLS) of the feature behaviors, and then the requirements validation is performed based on the generated formalisms. To our best knowledge, the main drawbacks of using NLS lie in: 1) ambiguities in the NLS cause imprecise definitions and even wrong understanding of the feature behaviors and, 2) the direct translation from the NLS to a FLS tends to be very costly and, 3) the NLS hinders to a large extent the possibility of performing automatic feature-oriented requirements validation.

To challenge the feature interaction problem and make up for the deficiency in the current practice, in this paper we propose a model-based approach to feature-oriented requirements validation. To be specific, our approach firstly specifies features by using an informal yet restricted natural language (from scratch) without losing ease of use, and then formalizes a set of executable models based upon the aforementioned intermediate specifications to perform the requirements validation. The approach is comprised of four steps as follows:

- **Feature Specification** specifies the behaviors and requirements of features by following the restricted use case modeling (RUCM) approach [12], which adopts a generic use case template and several restriction rules to reduce ambiguity and facilitate further analysis.
- **Feature Behaviors Formalization** formalizes the feature behaviors in terms of a formal yet literate specification using the extended Timed Abstract State Machine (eTASM) language [13], which generates executable models for analysis.
- **Feature Requirements Formalization** models the fea-

ture requirements by using the Observers technique [13] in eTASM for validation purpose.

- **Feature Validation** performs three kinds of model-based validation checking to detect the hidden flaws in the selected features, including *Logical Consistency Checking*, *Coverage Checking*, and *Model Checking*.

We also demonstrate the applicability of our approach through an illustration application, and the remainder of this paper is organized as follows: An introduction to the background knowledge is presented in Section II. Section III introduces the illustration application i.e., the Vehicle Locking-Unlocking (VLU) system. Our approach to feature-oriented requirements validation is described and illustrated by using the VLU system in Section IV. Section V discusses the related work, and finally concluding remarks and future work are drawn in Section VI.

II. BACKGROUND

In this section, we briefly introduce the RUCM approach [12] and the formal specification language eTASM [13] used in our approach for a better understanding.

A. Restricted Use Case Modeling

The restricted use case modeling (RUCM) [12] is a use case modeling approach that extends the UML [14] use case diagram by proposing a use case template and 26 restriction rules for reducing ambiguity and easing automated analysis. In our work, we specify features by populating the proposed use case template and following the restriction rules. In order to meet our needs, we make two slight modifications to the template. First, for the purpose of traceability, the use case name is required to follow the form of *FeatureName_UseCaseName* or merely *FeatureName*. Second, the *Basic Description* entry is replaced by *Feature Requirement* which specifies the requirement that the feature is associated with. Figure 1 shows the modified template and the brief explanation for each entry.

| | |
|---------------------|--|
| Use Case Name | In the form of <i>FeatureName_UseCaseName</i> or merely <i>FeatureName</i> . |
| Feature Requirement | Specifies the feature requirement. |
| Precondition | What should be true before the use case is executed. |
| Primary Actor | The actor who initiates the use case. |
| Secondary Actors | Other actors the system relies on to accomplish the functionality of the use case |
| Dependency | Include and extend relationships to other use cases. |
| Generalization | Generalization relationships to other use cases. |
| Basic flow steps | Specifies the main successful path in terms of a sequence of steps and a postcondition |
| | Steps (numbered) Flow of events |
| | Postcondition What should be true after the basic flow executes. |
| Specific Alt. Flow | Applies to one specific step of the reference flow |
| | RFS A reference flow step number where flow branches from. |
| | Steps (numbered) Flow of events |
| | Postcondition What should be true after the basic flow executes. |
| Bounded Alt.Flow | Applies to more than one step of the reference flow, but not all of them. |
| | RFS A list of reference flow steps where flow branches from. |
| | Steps (numbered) Flow of events |
| | Postcondition What should be true after the basic flow executes. |
| Global Alt.Flow | Applies to all the steps of the reference flow. |
| | Steps (numbered) Flow of events |
| | Postcondition What should be true after the basic flow executes. |

Fig. 1. The modified use case template of RUCM.

The feature behaviors are specified via use case flows, which are composed of one basic flow and one or more alternative flows. The basic flow specifies the main execution path in terms of a sequence of steps and a postcondition. Alternative flows specify execution branches when deviations occur somewhere in the reference flow that can be the basic flow or an alternative flow. There are three types of alternative flows: a specific alternative flow refers to a specific step in the reference flow; a bounded alternative flow refers to more than one step in the reference flow; a global alternative flow refers to all steps in the reference flow. RUCM defines 16 restriction rules to constrain the use of natural language, as shown in Figure 2. A set of keywords are also defined in the other 10 rules to specify control structures. For example, the keyword VALIDATES THAT (as shown in Figure 4) is used for condition checking. In particular, if the condition evaluates to be true, the current flow continues, otherwise an alternative flow will be executed. The detailed description of all the restriction rules and keywords are provided in [12].

| # | Description | Explanation |
|-----|---|---|
| R1 | The subject of a sentence in basic and alternative flows should be the system or an actor | Enforce describing flows of events correctly. These rules conform to our use case template (the five interactions). |
| R2 | Describe the flow of events sequentially | |
| R3 | Actor-to-actor interactions are not allowed | |
| R4 | Describe one action per sentence. (Avoid compound predicates.) | Otherwise it is hard to decide the sequence of multiple actions in a sentence. |
| R5 | Use present tense only | Enforce describing what the system does, rather than what it will do or what it has done. |
| R6 | Use active voice rather than passive voice | Enforce explicitly showing the subject and/or object(s) of a sentence. |
| R7 | Clearly describe the interaction between the system and actors without omitting its sender and receiver | Commonly required for writing UCs |
| R8 | Use declarative sentences only. "Is the system idle?" is a non-declarative sentence. | |
| R9 | Use words in a consistent way. | Keep one term to describe one thing |
| R10 | Don't use modal verbs (e.g., might) | Modal verbs and adverbs usually indicate uncertainty; therefore metrics should be used if possible |
| R11 | Avoid adverbs (e.g., very). | |
| R12 | Use simple sentences only. A simple sentence must contain only one subject and one predicate | Reduce ambiguity and facilitate automated natural language parsing. |
| R13 | Don't use negative adverb and adjective (e.g., hardly, never), but it is allowed to use not or no | |
| R14 | Don't use pronouns (e.g. he, this). | |
| R15 | Don't use participle phrases as adverbial modifier. | |
| R16 | Use "the system" to refer to the system under design consistently. | Keep one term to describe the system; therefore reduce ambiguity |

Fig. 2. The rules of RUCM to constrain the use of natural language.

B. The Extended TASM Language

eTASM [13] is a formal language for the specification of safety-critical systems, which extends the Timed Abstract State Machine (TASM) language [15] with the Observer and Event constructs. eTASM inherits the easy-to-use characteristic from TASM, which is a literate specification language understandable and usable without extensive mathematical training. An eTASM model consists of three parts – an environment, a set of machines, and a set of observers. The environment defines the set and the type of machine variables which machines can monitor or control, and the set of named resources which machines can consume. An machine consists of a set of monitored variables (which can affect the machine execution), a set of controlled variables (which machines can modify), and a set of machine rules. The set of rules specify the

machine execution logic in the form of “if *condition* then *action*”, where *condition* is an expression depending on the monitored variables, and *action* is a set of updates of the controlled variables. We can also use the rule “else then *action*” which is enabled merely when no other rules are enabled. A rule can specify the annotation of the time duration and resource consumption of its execution. The duration of a rule execution can be the keyword *next* that essentially states the fact that time should elapse until one of the other rules is enabled. The observers will monitor the events triggered by the execution of machines, and each observer represents one correctness property of interests that should be satisfied by the proposed system. In the eTASM language, four types of events can be triggered: The *ChangeEvent* type is triggered by a specific eTASM environment variable whenever its value is updated, the *ResourceUsedUpEvent* is triggered by the case whenever the resource of the application is consumed totally, and the *RuleEnableEvent* and *RuleDisableEvent* are triggered whenever the corresponding eTASM rule is enabled or disabled, respectively. An observer is made up of an *ObserverEnvironment*, a *Listener*, and an *Observation*. The *ObserverEnvironment* defines a set of observer variables and an *EventsFilter* that filters out irrelevant event types to the observer. The *Listener* specifies the expected events sequence following the syntax and semantics of *regular expression*. The *Observation* indicates the monitoring result, i.e., whether the correctness property monitored by the observer is satisfied.

eTASM describes the basic execution semantics as the computing steps with time and resource annotations: In one step, it reads the monitored variables, selects a rule of which *condition* is satisfied, consumes the specified resources, and after waiting for the duration of the execution, it applies the update set instantaneously. During the execution, eTASM triggers events whenever possible. The events sequence is monitored by observers. Once an expected sequence is observed, the corresponding monitoring result will be concluded. As a specification language, eTASM supports the concepts of parallelism (which stipulates that eTASM machines are executed in parallel) and hierarchical composition (which is achieved by means of auxiliary machines which can be used in other machines). There are two kinds of auxiliary machines - *function* machines that can take machine variables as parameters and return an execution result, and *sub* machines that can encapsulate machine rules for reuse purpose [15]. Communication and interaction between machines can be achieved by defining corresponding environment variables.

III. ILLUSTRATION APPLICATION

In this section, we describe a simplified Vehicle Locking-Unlocking (VLU) system, as a running example to illustrate our approach in this work. The proposed VLU system aims to replace the mechanical key, as a control access to a vehicle, and it follows a common pattern in feature-oriented requirements specification: The basic functionality is encapsulated as an individual feature, and additional/optional enhancements are specified as features that provide the increments in func-

tionality. Specifically, such features are Central Locking (CL), Auto-lockout (AUL) and Anti-lockout (ANL). Figure 3 shows the features of the VLU system in the form of technical feature model tree which is presented in EAST-ADL [16].

Central Locking (a basic feature) locks and unlocks all the doors of the vehicle upon receipt of a command from the user key fob.

Auto-lockout (an optional feature) locks all the doors of the vehicle when a timeout expires after the vehicle has stopped. It provides a theft protection in case that the driver forgets locking the doors manually.

Anti-lockout (an optional feature) enables unlocking of the doors while a key is in ignition after the vehicle has stopped, of which purpose is to prevent the driver from being locked out of the vehicle.

In simple applications such as the one above, it is possible to manually analyze the interactions between features for requirements validation. However, the real-world systems often have a large number of complex features, making the pertaining manual analysis extremely time-consuming and error-prone. The main motivation for our approach is to provide a semi-automatic technique for feature-oriented requirements validation for automotive systems, by performing undesirable feature interaction analysis. In the rest of the paper, we will use the aforementioned simplified application to illustrate our approach for features modeling, features specification, and auto-detection of undesired interactions.

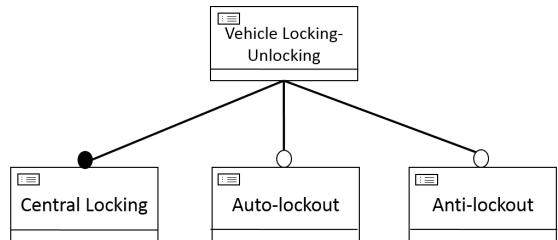


Fig. 3. The technical feature model tree of the VLU system.

IV. THE APPROACH TO FEATURE-ORIENTED REQUIREMENTS VALIDATION

In this section, we will introduce our approach that addresses the issue of formalizing and validating feature-oriented requirements specifications. In general, our approach is conducted in four steps as follows:

- **Step 1: Feature Specification** specifies the behaviors and requirements of features by using the RUCM use cases, which facilitates the further analysis.
- **Step 2: Feature Behaviors Formalization** formalizes the feature behaviors using the eTASM machines, which are executable analysis models.
- **Step 3: Feature Requirements Formalization** formalizes the feature requirements by using the eTASM Observer technique.

TABLE I
THE IDENTIFIED USE CASES AND CORRESPONDING FEATURE REQUIREMENTS.

| Feature | Use case name | Feature requirements |
|-----------------|---------------|--|
| Central Locking | CL_Lock | The system shall lock the doors |
| | CL_Unlock | The system shall unlock the doors |
| Auto-lockout | Auto-Lockout | The system shall automatically lock the doors after 20 seconds, when the vehicle has stopped |
| Anti-lockout | Anti-Lockout | The system shall anti-lock the doors if the key is in ignition and the vehicle has stopped |

| Use Case Name | CL_Lock | | |
|---------------------------------------|---|------------------|---------------|
| Feature Requirement | The system shall lock the doors of the vehicle | | |
| Precondition | None | | |
| Primary Actor | Key fob | Secondary Actors | Doors, Lights |
| Dependency | None | Generalization | None |
| Basic flow steps | 1) Key fob sends a "lock" command to the system. 2) The system VALIDATES THAT the doors are close. 3) The system locks the door. 4) The system flashes lights to indicate the completion of locking. | | |
| Postcondition | Doors are locked. Lights are off. The system is idle. | | |
| Specific Alt. Flow (RFS Basic flow 2) | 1) The system does nothing. Postcondition: Doors remain open. Lights are off. | | |
| Bounded Alt.Flow | None | Global Alt.Flow | None |

Fig. 4. The CL_Lock use case of the CL feature.

- **Step 4: Feature Validation** performs three types of checking by using model-based analysis techniques, to detect the hidden flaws in feature specifications.

We will go into details about each step by introducing the adhering sub-steps and show a running example to illustrate our approach. Specifically, Section IV-A introduces feature specification using the RUCM use cases. Section IV-B and Section IV-C discuss modeling of the behaviors and requirements of features respectively. Section IV-D presents the analysis and results of feature validation of the VLU system.

A. Feature Specification

The Feature Specification step describes the features of a system in a restricted natural language, which can facilitate the further transformation from an informal specification to the formal one, for the purpose of validation. In this step, each feature will be specified by following the RUCM approach, and there are two sub-steps in our work as follows:

- **Step 1.1: Use Cases Identification:** Since a feature captures a set of cohesive functionalities in the form of requirements and corresponding behaviors, it is therefore necessary to split the functionalities and identify the possible use cases based on the expert's understanding of the feature.
- **Step 1.2: Use Cases Specification:** Use cases are specified by filling the RUCM template using a restricted natural language. In order to facilitate the further analysis, some predefined restriction rules must be followed.

In the VLU system, there are three selected features i.e., *CL*, *AUL* and *ANL* (as introduced in Section III). The specification of features, as an example, is illustrated by applying the proposed steps to the *CL* feature. To be specific, since the *CL* feature describes two opposite functionalities, two use cases can be thereby identified in terms of *CL_Lock* and *CL_Unlock*, as shown in Table I. Figure 4 and Figure 5 describe the filled use case templates of *CL_Lock* and *CL_Unlock*, respectively.

B. Feature Behaviors Formalization

This step is to analyze the specified RUCM templates and formalize the corresponding feature behaviors by using eTASM models which are executable simulation models for analysis. The Feature Behaviors Formalization step contains four sub-steps:

| Use Case Name | CL_UnLock | | |
|---------------------|---|------------------|---------------|
| Feature Requirement | The system shall unlock the doors of the vehicle | | |
| Precondition | None | | |
| Primary Actor | Key fob | Secondary Actors | Doors, Lights |
| Dependency | None | Generalization | None |
| Basic flow steps | 1) Key fob sends an "unlock" command to the system. 2) The system unlocks the door. 3) The system flashes lights to indicate the completion of unlocking. Postcondition: Doors are close but unlocked. Lights are off. The system is idle | | |
| Specific Alt. Flow | None | | |
| Bounded Alt.Flow | None | Global Alt.Flow | None |

Fig. 5. The CL_Unlock use case of the CL feature.

- **Step 2.1: System Constituents Identification** extracts the relevant system constituents referred in the RUCM use cases and specifies them in eTASM machines.
- **Step 2.2: Constituents Interaction Identification** identifies the interactions between different system constituents referred in the RUCM use cases and specifies them in eTASM environment variables.
- **Step 2.3: Machine Rules Specification** analyzes the possible states of identified machines and specifies feature behaviors by using a set of eTASM machine rules.
- **Step 2.4: Property Annotation** adds non-functional property annotations to the relevant eTASM machines.

1) *System Constituents Identification:* The identification of the system constituents from the use cases is of importance in the process to formalize the behaviors of the proposed system and model the scenarios for model-based validation. In order to do so, we recommend the following two tasks:

- *External Constituents Identification:* Use case actors are considered as external constituents which interact with the proposed system. The external constituents will be modeled to simulate the execution scenarios.
- *Internal Constituents Identification:* Each use case is considered to be an internal constituent, making up the proposed system. The internal constituents will be modeled to simulate the proposed system.

In this step, a list of eTASM machines w.r.t. the identified constituents is defined for the VLU system, as shown in Table II.

2) *Constituents Interaction Identification:* Two types of interaction between the sending constituent (i.e., sender) and receiving constituent (i.e., receiver) are considered in our

TABLE II
THE ETASM MACHINES IDENTIFIED FOR THE VLU SYSTEM.

| Machine | Quantity | Category | Brief Description |
|--------------------|----------|----------|--|
| KEY_FOB | 1 | External | model the key fob's behavior |
| LIGHT | 1 | External | model the light's behavior |
| DOORS | 1 | External | model the behavior of doors |
| IGNITION | 1 | External | model the behavior of ignition |
| VEHICLESPEEDSENSOR | 1 | External | model the behavior of vehicle speed sensor |
| CL_LOCK | 1 | Internal | lock doors |
| CL_UNLOCK | 1 | Internal | unlock doors |
| AUTO_LOCKOUT | 1 | Internal | lock doors when timeout expires |
| ANTI_LOCKOUT | 1 | Internal | anti-lock doors if key is in ignition |

approach:

- *Data Transmission Interaction (DTI)* represents that data (such as the state information and various sensor values) are transferred from the sender to the receiver, which are modeled as eTASM environment variables. The variables are named as *sender_datatype* which denotes the transferred data. Line 2 in Figure 7 shows an example.
- *Data Modification Interaction (DMI)* represents that the data of the receiver is directly changed by the sender, which are modeled via directly modifying the value of the receiver's environment variable. The variables are named in the form of *receiver_datatype*, which denotes the modified data. One example can be found in Lines 9 and 10 in Figure 7.

Since RUCM requires the interaction between a system and an actor to be clearly described without omitting some information about its sender and receiver, it is therefore easy to identify interactions between constituents from the use case models. Figure 6 shows the identified interactions in our VLU system, which are twelve interactions. Further, the solid lines represent DTIs, and the dashed lines represent DMIs.

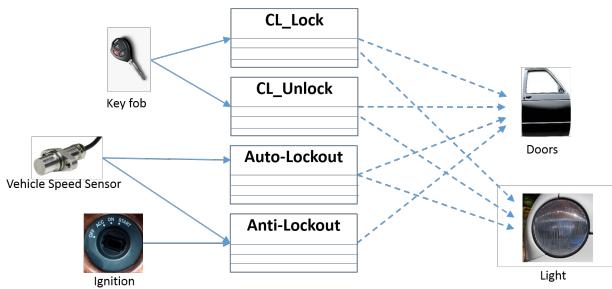


Fig. 6. The identified interactions between constituents in the VLU system.

3) *Machine Rules Specification*: The restricted use case flow sentences (e.g., in basic and/or alternative flows) can to a large extent facilitate the transformation from use case models to analysis models [17]. Based on the sentences specified in use case flows, we recommend the following sub-steps to specify the eTASM machine rules:

- *Identification of possible states of the corresponding constituent*: The possible states of a constituent can be identified via analyzing the adjectives and verbs in the use case flows. A user-defined type is used to represent the

possible states, and a state variable is defined to denote the current state of the constituent.

- *Identification of the transition conditions of states*: The conditions of a certain machine rule are given, according to the pertaining values of the state variables and the transition conditions.
- *Identification of the actions when the system enters a specific state*: The actions of machine rules are specified, based on the behaviors of a constituent and the next possible state.

In our VLU application, there are five external constituents and four internal constituents under consideration, as shown in Table II. The KEY_FOB machine simulates the behaviors of a user's key fob. This machine has two possible states *lock* and *unlock*, in which the *lock/unlock* state denotes that the lock/unlock command is sent to the proposed VLU system. The LIGHT machine simulates the behaviors of the vehicle light, which has two states i.e., the *flashed* state and the *off* state. The *flashed* state denotes that the light flashes for several times. The DOORS machine simulates the behaviors of the vehicle doors, which has three possible states *open*, *close* and *locked*. The IGNITION machine simulates the behaviors of the vehicle ignition, which has two possible states *haskey* and *nokey*. The VEHICLESPEEDSENSOR machine simulates the behaviors of the vehicle speed sensor, which has two possible states *still* and *running*.

The CL_LOCK machine, as shown in Figure 7, models the CL_LOCK use case. The machine has two possible states in terms of *idle* and *lockdoor*. The Rule *ReceiveCommand* represents that the system receives the lock command from the key fob. The Rule *Locking* represents that the system locks the doors. The Rule *Idle* keeps the machine alive and represents the system is idle. The CL_UNLOCK machine has similar rules, which will not be introduced for simplicity.

The AUTO_LOCKOUT machine, as shown in Figure 8, models the behaviors of the Auto-Lockout use case. The Rule *Timeout* represents that when the vehicle stops and the doors are close, the feature will be activated upon the timeout expires (i.e., 20s in our case). The Rule *Autolock* represents that the system is to automatically lock the doors. The Rule *Timer* represents the timer measuring time intervals. Rule *Idle* keeps the machine alive and represents that the timer will be reset either when the doors are open or when the vehicle starts running.

```

1 R1:ReceiveCommand{
2   if cclock_state = idle and keyfob_cmd = lock then
3     cclock_state := lockdoor;
4     keyfob_cmd := NONE;
5   }
6 R2:Locking{
7   t:=locking_time;
8   if cclock_state = lockdoor then
9     door_state := locked;
10    light_state := flashed;
11    cclock_state := idle;
12  }
13 R3:Idle{
14   t := next;
15   else then
16     skip;
17 }

```

Fig. 7. The eTASM main machine models the CL_Lock use case.

```

1 R1:Timeout{
2   if aul_state = idle and door_state = close and
3     vehicle_state = still and timer = 20 then
4     aul_state := timeout;
5     timer := 0;
6   }
7 R2:Autolock{
8   t:=locking_time;
9   if aul_state = timeout then
10    door_state := locked;
11    light_state:= flashed;
12    aul_state := idle;
13  }
14 R3:Timer{
15   t := 1;
16   if aul_state = idle and door_state = close and
17     vehicle_state = still and timer < 20  then
18     timer := timer + 1;
19  }
20 R4:TimerReset{
21   t := next;
22   else then
23     timer := 0;
24 }

```

Fig. 8. The eTASM main machine models the Auto-Lockout use case.

The ANTI_LOCKOUT machine, as shown in Figure 9, models the behaviors specified in the Anti-Lockout use case. The Rule *HasKey* represents that when the vehicle stops, the feature will be activated if the key is in ignition. The Rule *Antilock* represents that the system is to unlock the doors after activated. The Rule *Idle* keeps the machine alive and represents that the system is idle.

4) *Property Annotation*: Validation of non-functional requirements in this stage relies on the estimates of the pertaining non-functional properties of the proposed system. This step can be carried out in the following ways:

- The properties are determined based upon the non-functional requirements specified in the use cases.
- The properties are determined by using the experience or analysis of existing systems (in which estimates can be obtained by using existing well-known analysis methods, e.g., Worst-Case Execution Time (WCET) analysis [18], [19] for time duration of rules).

We annotate the aforementioned eTASM models with time durations, as shown in Figures 7, 8, and 9. The annotation

```

1 R1:HasKey{
2   if anl_state = idle and ignition_state = haskey and
3     vehicle_state = still then
4     anl_state := antilock;
5   }
6 R2:Antilock{
7   t:=unlocking_time;
8   if anl_state = antilock then
9     door_state := close;
10    anl_state := idle;
11  }
12 R3:Idle{
13   t := next;
14   else then
15     skip;
16 }

```

Fig. 9. The eTASM main machine models the Anti-Lockout use case.

terms *locking_time* and *unlocking_time* are either a specific value or a range of values.

C. Feature Requirements Formalization

Our approach proceeds with the formalization of feature requirements by using the eTASM Observer technique, which consists of four sub-steps as follows:

- **Step 3.1: Listener Specification** specifies the possible events sequence which represents the proposed system's observable functional behaviors and/or non-functional properties required by the feature requirements, and the corresponding actions taken on observer variables when the sequence is caught by a Listener.
- **Step 3.2: Observation Specification** formalizes a predicate depending on the observer variables. If the predicate of the Observation holds, i.e., evaluates to be true, it implies that the property satisfaction of the feature is achieved, as it can be observed in the proposed system.
- **Step 3.3: Events Filtering** identifies the interesting events and filters out the irrelevant events by specifying *EventsFilter*.
- **Step 3.4: Traceability Creation** links a specific Observer to the textual requirements. The link is used for requirements traceability from the formalization to natural language requirements in order to perform coverage checking.

In the VLU system, there are four feature requirements, i.e., *CL_Lock*, *CL_Unlock*, *Auto_Lockout* and *Anti_Lockout*. The specification of an observer is illustrated by applying the proposed steps to the *ANL* feature requirement. To be specific, the *ANL* feature requirement states "The system shall anti-lock the doors if the key is in ignition and the vehicle has stopped", and the interesting events sequence consists of three parts. The first part "*ANTI_LOCKOUT*→*HasKey*→*RuleEnableEvent*" denotes that the event is triggered when the Rule *HasKey* of the *ANTI_LOCKOUT* machine is enabled, modeling the behavior that the key is in ignition. The second part "[[^](*AUTO_LOCKOUT*→*Autolock*→*RuleEnableEvent* | *CL_LOCK*→*Locking*→*RuleEnableEvent*)]*" represents an arbitrary number of events that are not triggered by the enabling of either the Rule *Autolock* of the *AUTO_LOCKOUT*

machine or the Rule *Locking* of the CL_LOCK machine. Both of these two rules model the behavior that the doors are locked. The last part "ANTI_LOCKOUT->Idle->RuleEnableEvent" represents the event that is triggered when the Rule *Idle* of the ANTI_LOCKOUT machine is enabled, which models the situation in which the key is removed. If the events sequence is detected, the Observation "ov == true" evaluates to be true, which indicates the situation in which after the key is in ignition, the doors are not locked before the key is removed, i.e., the *ANL* feature requirement is satisfied in the eTASM model.

```

1 ObserverVariables:{  
2     Boolean ov := false;  
3 }  
4 EventsFilter:{  
5     filter out: ChangeValueEvent, ResourceUsedUpEvent,  
6             RuleDisableEvent;  
7 }  
8 Listener:{  
9     listening ANTI_LOCKOUT→HasKey→RuleEnableEvent  
10        [^(AUTO_LOCKOUT→Autolock→RuleEnableEvent |  
11          CL_LOCK→Locking→RuleEnableEvent)]  
12        ANTI_LOCKOUT→Idle→RuleEnableEvent then  
13        ov := true;  
14 }  
15 Observation:{  
16     ov == true;  
17 }
```

Fig. 10. The observer for the ANL feature requirement.

D. Feature Validation

Validation of the formalized requirements aims at increasing the confidence of the validity of selected features. In this work, we assume that there is a semantic equivalence relation between the RUCM use cases and eTASM models. This is built upon the fact that the eTASM models are derived, by following the proposed modeling steps as well as our thorough understanding of the VLU system. The validation goal is achieved by following several analysis steps, based on the use of the derived eTASM models which may help to pinpoint flaws that are not trivial to detect. Such validation steps in our approach are:

- **Step 4.1: Logical Consistency Checking.** The term of logical consistency can be intuitively explained as "free of contradictions in the specifications". In our work, the logical consistency checking is performed on the executable eTASM models, by using our developed tool TASM TOOLSET. Furthermore, there are two kinds of inconsistency flaws to discover. One kind of flaws is that two rules in the same machine are enabled simultaneously, which is usually caused by the fact that there exist unpredictable behaviors in the specification of the corresponding feature. The other is that different values are assigned to the same variable simultaneously by different machines, which is usually caused by the fact that there exist hidden undesirable feature interactions in the specifications of the corresponding features.

- **Step 4.2: Coverage Checking.** The coverage checking corresponds to checking whether the feature requirements can be observed in the integrated feature specifications, which is an important activity of requirements completeness checking. To perform the coverage checking, all the feature requirements are translated into observers which monitor the execution of the features specifications, i.e., the derived eTASM models. If an *Observation* cannot hold, it indicates that although the features specifications satisfy their individual requirements in isolation, there are behavioral inconsistencies in the integrated feature specification.

- **Step 4.3: Model Checking.** The eTASM machines can be easily translated into timed automata through the transformation rules defined in [15]. The transformation enables the use of the UPPAAL model checker to verify the various properties of the eTASM model. This type of checking aims at verifying whether the eTASM model is free of deadlock and whether an expected property specified in a feature requirement is satisfied by the eTASM model. It is necessary to stress that the essential difference between *Model Checking* and *Coverage Checking* is whether a property is exhaustively checked against a model or not. Although a sound property checking is desired, in some cases *Model Checking* will encounter state explosion problem, which limits its usefulness in practice.

We follow the aforementioned validation steps to check the validity of the selected features of the VLU system. First, we use the TASM TOOLSET to perform *Logical Consistency Checking* on the formalized eTASM model. Two inconsistencies are detected, one of which is that the Rule *Autolock* of the AUTO_LOCKOUT machine and the Rule *Antilock* of the ANTI_LOCKOUT machine update the *door_state* variable simultaneously with different values. An analysis of the inconsistency reveals: When the key is in ignition, the ANL feature will keep the doors unlocked. Meanwhile, if the autolock timeout expires, the AUL feature will try to lock the door. Since no rules are explicitly specified in the selected features to handle this situation, undesirable behaviors will occur. The other inconsistency is detected in a similar situation where the Rule *Locking* of the CL_LOCK machine and the Rule *Antilock* of the ANTI_LOCKOUT machine update the *door_state* variable simultaneously with different values. In this work, we correct such inconsistencies by assigning a higher priority (as an extra condition of the corresponding rule) to the Rule *Antilock*, which guarantees that it will be executed at first when both of two rules are enabled at the same time. Note that there are some other methods that can be used to remove the discovered inconsistencies, which are however out of the scope of this paper.

After the removal of the inconsistencies, we proceed to *Coverage Checking*. The TASM TOOLSET is applied, and the result has shown that the observations of all eTASM observers are met. Therefore, the integrated features specifications sat-

isfies the feature requirements, from the *Coverage Checking* perspective.

On the note about *Model Checking*, we first translate the eTASM model into timed automata, and then check the deadlock property as well as the feature requirements via UPPAAL. The corresponding results are: 1) *Deadlock free is satisfied* and, 2) the *CL_Lock feature requirement is satisfied* and, 3) the *CL_Unlock feature requirement is satisfied* and, 4) the *AUL feature requirement is satisfied* and, 5) the *ANL feature requirement is satisfied*. As a result, the satisfaction of deadlock-free and feature requirements has been achieved.

In summary, our approach has found *two* behavioral inconsistencies in the integrated features specifications. Although the VLU system is not complex, it is enough, as an illustrative example, to show how to perform feature-oriented requirements validation by following our proposed approach.

V. RELATED WORK

Kimbler et al. [20] introduce a user-oriented approach to feature interaction analysis. It aims first at creating use case models to describe different possible ways of using the system services, and then building service usage models which simulate the dynamic relations between services. This work is quite similar with our idea, however its focus is in the telecommunication domain. Moreover, we use the RUCM approach to facilitate the transformation from use case models to subsequent formalisms. Eriksson et al. [21] propose a software product line use case modeling approach i.e., PLUSS to modeling SPL. The difference between their work and ours is the purpose of using use cases. PLUSS aims to utilize use cases to capture variants of SPL, while our approach utilizes use cases to specify behavioral specifications of features. The white paper of EAST-ADL [16] mentions that use cases can be used to specify features but no more details were given. In this work, we have provided a set of steps to specify features and perform requirements validation.

Amyot et al. [10] propose an approach to detecting feature interactions of telecommunication systems, by using Use Case Maps (UCMs) for designing features, and LOTOS for the formal specification of features. Sampath et al. [22] present a formal specification and analysis method for automotive features in the early stages of software development process. This method starts with an empty specification, and then incrementally adds clauses to the specification until all the feature requirements are satisfied. Arora et al. [7] propose a method and algorithms for identifying and resolving feature interactions in the early stages of the software development life-cycle. The work uses *State Machines* to model the behavior of independent features, context diagrams to integrate independent features, and Live Sequence Charts to capture the interactions of features.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a novel approach to feature-oriented requirements validation by using the RUCM approach and the eTASM language. Our approach 1) specifies the

behaviors and requirements of features in the RUCM use cases and, 2) transforms such RUCM use cases to the formal yet literate eTASM models and, 3) performs the requirements validation by using the TASM TOOLSET and the model checker UPPAAL. Our illustration application using a Vehicle Locking-Unlocking (VLU) system has shown that our approach can achieve the goal of feature-oriented requirements validation via *Logical Consistency Checking*, *Coverage Checking*, and *Model Checking*.

As inspired by Scandurra et al. [17] showing the promise of rule-based transformation from RUCM use cases to analysis models, we will in the future combine the proposed modeling approach with such rule/pattern-based algorithms, to achieve a fully automatic transformation between the RUCM use cases and eTASM models. We are also interested in integrating our approach and related tools for the development of correct-by-construction systems (e.g., developed by EAST-ADL language) in a seamless and cost efficient way. Another part of future work also includes a wider industrial validation of our approach, as well as the improvement of our current TASM TOOLSET.

REFERENCES

- [1] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, “Feature-oriented domain analysis (foda) feasibility study,” Tech. Rep., Nov 1990.
- [2] J. Bosch, *Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach*. New York, NY, USA: Addison-Wesley Professional, May 29, 2000.
- [3] A. Classen, P. Heymans, and P.-Y. Schobbens, “What’s in a feature: A requirements engineering perspective,” in *Proceedings of FASE’08/ETAPS’08*, 2008, pp. 16–30.
- [4] D. Benavides, P. Trinidad, and A. Ruiz-Cortés, “Automated reasoning on feature models,” in *Proceedings of CAiSE’05*. Springer-Verlag, 2005, pp. 491–503.
- [5] X. Peng, W. Zhao, Y. Xue, and Y. Wu, “Ontology-based feature modeling and application-oriented tailoring,” in *Proceedings of ICSR’06*, 2006, pp. 87–100.
- [6] M. Mendonca, A. Wkasowski, and K. Czarnecki, “Sat-based analysis of feature models is easy,” in *Proceedings of SPLC’09*, 2009, pp. 231–240.
- [7] S. Arora, P. Sampath, and S. Ramesh, “Resolving uncertainty in automotive feature interactions,” in *Proceedings of RE’12*, Sep 2012, pp. 21–30.
- [8] A. F. Layouni, K. J. Turner, and L. Logrippo, “Conflict detection in call control using firstorder logic model checking,” in *Proceedings of ICFI’07*, 2007.
- [9] S. Apel and C. Kstner, “An overview of feature-oriented software development,” 2009.
- [10] D. Amyot, L. Charfi, N. Gorse, T. Gray, L. Logrippo, J. Sincennes, B. Stepien, and T. Ware, “Feature description and feature interaction analysis with use case maps and lotos,” in *Proceedings of FIW’00*, 2000, pp. 274–289.
- [11] M. Poppleton, “Towards feature-oriented specification and development with event-b,” in *Proceedings of REFSQ’07*, 2007, pp. 367–381.
- [12] T. Yue, L. C. Briand, and Y. Labiche, “A use case modeling approach to facilitate the transition towards analysis models: Concepts and empirical evaluation,” in *Proceedings of MODELS’09*, 2009, pp. 484–498.
- [13] J. Zhou, Y. Lu, and K. Lundqvist, “A tasm-based requirements validation approach for safety-critical embedded systems,” in *Proceedings of Ada-Europe’14*, June 2014.
- [14] O. M. Group, “OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2,” Tech. Rep., Nov. 2007.
- [15] M. Ouimet, “A formal framework for specification-based embedded real-time system engineering,” Ph.D. dissertation, Department of Aeronautics and Astronautics, MIT, 2008.

- [16] H. Blom, H. Lönn, F. Hagl, Y. Papadopoulos, M.-O. Reiser, C.-J. Sjöstedt, D.-J. Chen, and R. T. Kolagari, “EAST-ADL - An Architecture Description Language for Automotive Software-Intensive Systems,” The EAST-ADL 2 Consortium, Tech. Rep., 2012.
- [17] P. Scandurra, A. Arnoldi, T. Yue, and M. Dolci, “Functional requirements validation by transforming use case models into abstract state machines,” in *Proceedings of SAC’12*. NY, USA: ACM, 2012, pp. 1063–1068.
- [18] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, “The worst-case execution-time problem—overview of methods and survey of tools,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, pp. 1–53, April 2008.
- [19] Y. Lu, “Pragmatic Approaches for Timing Analysis of Real-Time Embedded Systems,” Ph.D. dissertation, Mälardalen University, June 2012.
- [20] K. Kimbler and D. S. birk, “Use case driven analysis of feature interactions,” in *Feature Interactions in Telecommunications Systems, IOS*, 1994, pp. 167–177.
- [21] M. Eriksson, J. Börstler, and K. Borg, “The pluss approach: Domain modeling with features, use cases and use case realizations,” in *Proceedings of SPLC’05*, 2005, pp. 33–44.
- [22] P. Sampath, S. Arora, and S. Ramesh, “Evolving specifications formally,” in *Proceedings of RE’11*, Aug 2011, pp. 5–14.

Product Knowledge Configurator for Requirements Gap Analysis and Customizations

Preethu Rose Anish

Tata Research Development and Design Centre (TRDDC)
Tata Consultancy Services
Pune, India
preethu.rose@tcs.com

Smita Ghaisas

Tata Research Development and Design Centre (TRDDC)
Tata Consultancy Services
Pune, India
smita.ghaisas@tcs.com

Abstract—Product knowledge plays an important role in identifying the requirements of the desired variant and configuring the existing product to the present needs of a customer. The success of a product-based business depends to a great extent on how efficiently and accurately the existing product knowledge is utilized for customization needs. Oftentimes however, product knowledge resides with few key individuals in an organization. In the absence of their involvement, project teams may redevelop product features unnecessarily, resulting in an effort overhead. Such overdependence poses a risk to projects. To identify the requirements for the variants accurately and efficiently, we need to have a thorough knowledge of the existing product features. In this paper, we discuss our work on representing product knowledge and reusing it in a Requirements Engineering (RE) exercise for a large project involving product customization. We present our experience from using the configurator for requirements gap analysis and customizations.

Index Terms—Requirements configurator, Ontology, Product Line, Financial product suite, Configuring products, Knowledge assisted configuration.

I. INTRODUCTION

The success of product-based business depends to a great extent on the clarity of customer requirements [1, 2, and 3]. Product configuration is the process of generating a variant from an existing product suite and additional product specifications for a given variant [22]. For an efficient product configuration; customer requirements have to be captured, analyzed, and compared accurately with the existing product variants in the product suite. Product knowledge plays an important role in this respect [1, 4, and 5]. Gap analysis is defined as a method by which customer requirements are compared with the existing product feature specifications in the product suite to arrive at a match or mismatch.

Requirements gap analysis is a core activity in a product configuration and customization exercise. A thorough knowledge of the existing product features in the product suite is essential to accurately identify the requirements for a new product. We can use the knowledge of the existing product features to compare it with the requirements of the new product, perform a gap analysis and use the analysis in customization of the product. In practice, gap analysis is often

a manual activity; involving workshop sessions with stakeholders (for instance JAD sessions [14]). Also, since the knowledge resides with a few key individuals in an organization; teams depend heavily on these people. This presents a big risk associated with loss of knowledge if these people leave the organization. In the absence of their involvement, product teams may redevelop already existing features. This not only results in effort-overhead but also makes the whole exercise of requirement gap analysis subjective.

Our interactions with the product experts in our company revealed that manual analysis can unearth only the easily visible and accessible high level details in the product variants. At the high level, all products look similar and hence, during gap analysis, the products can be misconceived to be 100 % similar. The gap analysis done in the absence of adequate product knowledge entails a high probability of leaving out on the unobvious lower level product details. In the product hierarchy, it is often at the lower levels that the actual gaps exist. The unavailability of knowledge-assistance to uncover the deeper level details can make the whole exercise of gap analysis subjective and susceptible to errors and omissions. These concerns are representative of today's business scenario wherein there is a progressive shift from the traditional one-size-fits-all products to more customer-specific products [8-11].

In an earlier publication, we discussed our work on a product knowledge configurator to represent product knowledge so that it can be reused in RE exercises for large projects in the same domain [7]. The approach involves knowledge-assisted product requirements evolution from a knowledge base (KB). The KB consists of product knowledge elements such as features (risk-cover pairs), attribute groups, attributes, business rules and the associated business processes, sub processes and stakeholders. In collaboration with the customers and domain experts, the requirement analysts can use the configurator to perform requirement gap analysis.

We have used the configurator to conduct RE, specifically for gap analysis and configuration of product specifications in a large distributed Insurance domain in Europe.

The rest of the paper is organized as follows: section II details the solution approach. Section III outlines the case study while section IV presents discussion and conclusion

II. PRODUCT KNOWLEDGE CONFIGURATOR

In this section, we elaborate the ontology-based product knowledge configurator in detail. The conceptual model of the configurator has been introduced in our previous publication [7].

The ontologies are organized along three contexts: (1) Product Knowledge (2) Environment (3) Business domain. We use Product Knowledge Ontology to represent the Insurance product and its variants. It captures the product primitives in terms of (1) concepts such as features (risk-cover pairs), attribute groups, attributes, business rules and (2) associations between the concepts. Additionally, we capture associated business processes, sub processes and stakeholders as well. Table I provides some definition along with examples, in the parlance of the product under discussion.

TABLE I. PRODUCT KNOWLEDGE ONTOLOGY CONCEPTS AND EXAMPLES

| Concept | Definition | Example(s) |
|------------------|--|--|
| Risk | Risk is the potential of losing something of value | <i>Permanent disability</i> |
| Cover | Cover describes the situations you are insured against. | <i>Cover against permanent disability</i> |
| Attribute Group | Attribute group is a set of logically coherent attributes | <i>Insured details</i> |
| Attribute | An attribute is a quality or feature regarded as a characteristic or inherent part of someone or something [20] | <i>For insured details, the attributes are: insured name, age, gender, occupation etc.</i> |
| Business Rule | Business rules are external constraints that are not under the control of the organization. | <i>Laws of land, domain specific policies</i> |
| Business Process | A business process is a collection of related, structured activities or tasks that produce a specific service or product for a particular customer or customers [21] | <i>Claims process, Litigation process, Adjudication process</i> |
| Sub Process | A sub process is a collection of related, structured activities or tasks within a process | <i>For the process Claim; Claims intimation, Claim booking are sub processes</i> |
| Stakeholder | Stakeholders are entities that are directly or indirectly impacted by the product under consideration | <i>Insured, Insurer</i> |

The product knowledge is expressed and organized as an instance of the model. The Product Knowledge Ontology interacts with two other ontologies:

- **Environmental Context Ontology** which is used to capture the environmental parameters such as Domain (e.g. *Insurance*), Business (e.g. *Life*), Line of Business (e.g. *Group Life*), Product Line (e.g. *Personal*), Geography (e.g. *Europe*) and Company (e.g. *ABC Inc.*). These abstractions allow for representation of knowledge of the environment in which the product is to be deployed. This is important as customers may need to conform to environmental specific laws and may have different organizational policies. Thus, selection of the environmental parameters ensures that only the relevant product specification is made available to the requirement analyst and the analyst's view is not cluttered with any irrelevant information.
- **Business Domain Ontology** which is used to capture the intricacies of a given domain such as Insurance and therefore contains business concepts, their relationships and constraints. For example, consider the following scenario from Insurance domain – *For group life claims, in the event of the occurrence of a physical disability; the policyholder may submit a claim request.* The abstractions such as **BusinessEvent** (*Occurrence of physical disability*), **Party** (*Policyholder*), and **BusinessAction** (*Submit a claim request*) are used to capture this information.

Figure 1 show examples of the three ontologies depicted using the UML class diagram notation. It should be noted that we have shown only example concepts from the Business Domain and the Environmental Context Ontology to explain their mappings with the Product Knowledge Ontology. A few examples of mappings between the three ontologies are as follows:

- The **BusinessEvents** (e.g., *Policy Purchase*), **BusinessActions** (e.g. *Scrutinize Details*), and **BusinessDecisions** (e.g., *Policy Issuance*) in the Business Domain Ontology are together represented as **BusinessProcess** (e.g., *New Business*) in the Product Knowledge Ontology.
- **BusinessConstraint** (e.g. *Legislation#123*) in the Business Domain Ontology maps to **BusinessRule** (e.g., *Rule 1*) in Product Knowledge Ontology.
- **BusinessValue** (e.g. *Profit Margin*) in the Business Domain Ontology maps to **BusinessGoal** (e.g. *Optimize premium payout*) in Product Knowledge Ontology.
- **BusinessDocument** (e.g. *Vehicle Policy*) in the Business Domain Ontology maps to **Policy** (e.g. *ABC Vehicle Policy*) in Product Knowledge Ontology.

The semantic assistance is achieved by employing the “bridge classes” and inference rules written in the Semantic Web Rule Language (SWRL) [15]. The bridge classes specify

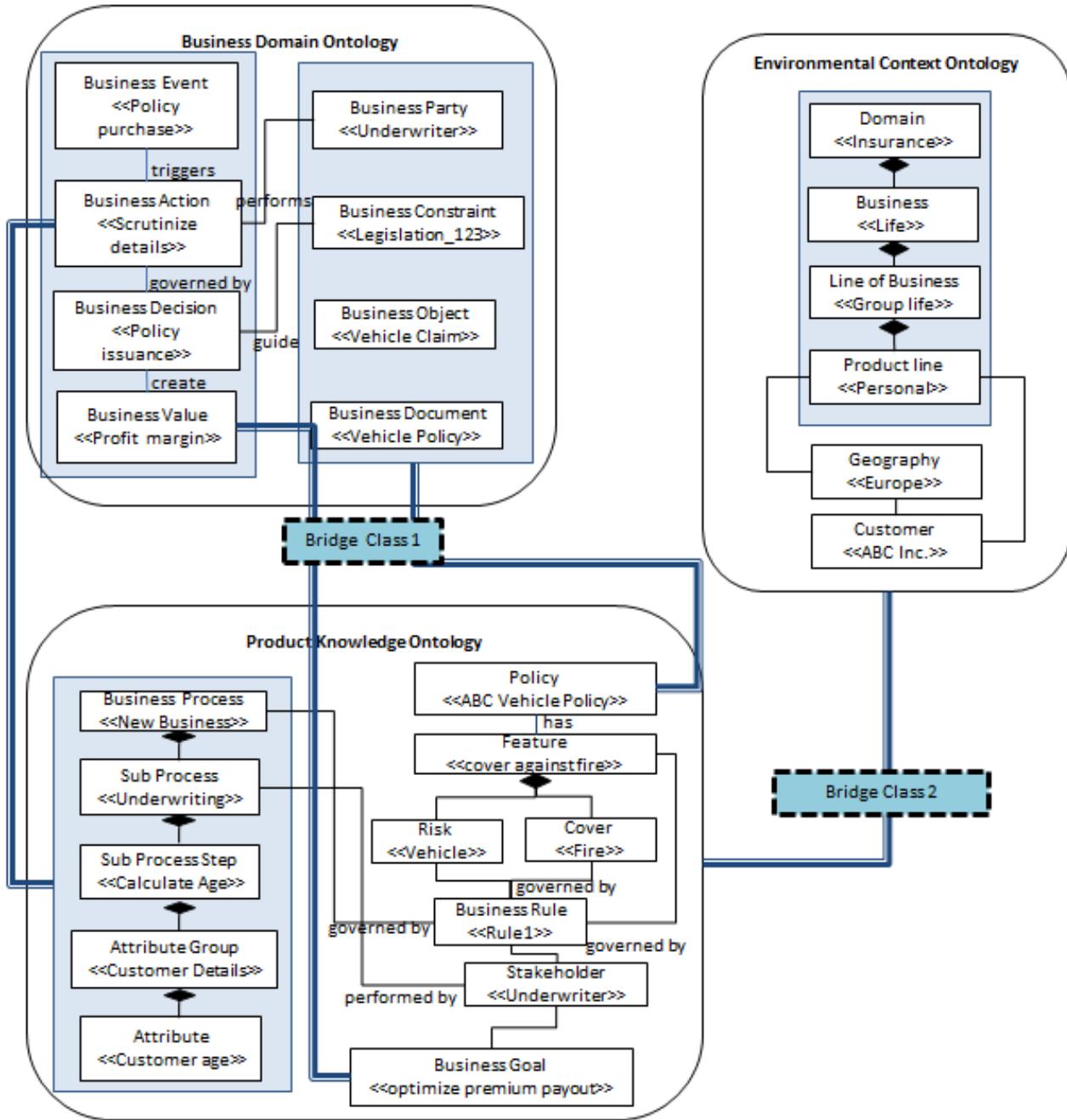


Fig. 1. Example instances of ontologies and bridge classes that enable context-specific recommendations

semantic mappings between the three ontologies. We define rules that refer the ontology instances and provide recommendations based on the integrated inference. The recommendations may be specific to a singular ontology or may span across the three ontologies when necessary, in response to the actions of the business analyst. For example, if the requirement analyst selects a set of environmental parameters specific to the customer such as **Domain** (e.g. *Insurance*), **Business** (e.g. *Life*), **Line of Business** (e.g. *Group Life*), **Product Line** (e.g. *Personal*), **Geography** (e.g. *Europe*) and **Company** (e.g. *ABC Inc.*), the product knowledge

configurator would present her a suitable product variant (based on the selected parameters) from previously executed projects, if already available in the repository. Else, a generic specification is presented. The customer representative and the analyst would review this together to determine the point of departure (whether from a product variant or from the generic specification) for product configuration. The customer would then suggest additions /modifications as per their needs. The analyst would begin the process of configuring the generic or the selected variant of the product specification from the product suite to incorporate the customer's inputs. At this

stage, the configurator would create a copy of the generic or the selected variant of the product specification and make it available in the workspace designated for the requirement analyst. As the analyst incorporates customer's inputs to create a variant of the product, she receives context-specific recommendations from the product knowledge configurator. The recommendations are facilitated by the underlying ontologies and the semantic rules that operate on it. For instance, when the representative adds a new **Cover** (*Permanent Disability*) upon suggestion by the customer, the configurator alerts her that the cover that she has added already exists in some other configured product (e.g. *Product A*) in the product suite. She is recommended to look up the associated artifacts with this **Cover** (*Permanent Disability*) in *Product A* such as the associated **Rules** (*A claim against permanent disability should be intimated within 1 week of its occurrence*), **Processes** (*Claim Process*), **Sub processes** (*Claim Registration*), **Attributes** (*Policy Id.*) etc. When she selects the suggested rule (*A claim against permanent disability should be intimated within 1 week of its occurrence*), she receives an alert to verify if this rule contradicts with some other rule (*All claims must be intimated within 30 days of its occurrence*) that already exists in the selected product specification. Similarly, if she modifies a **Sub Process** (e.g. *claim Registration*), she is alerted about the impacted artifacts such as **Sub process step** (e.g. *Capture claim details*), **Attribute Group** (e.g. *Details for Claim intimation*), **Attribute** (e.g. *Cause of disability*). Upon completing the configuration as detailed above, the analyst can produce a gap analysis report from the configurator. The report highlights the 'evolution' of the product specification from its generic version to its configured variant. This information is critical to evaluate the amount of further customization that may be needed (if any).

In section III, we evaluate the product knowledge configurator as opposed to the traditional manual technique to identify the gaps that exist between the baseline version and the configured version of the product specification.

III. EVALUATION IN INDUSTRIAL SETTING

In this section, we present results of using the product knowledge configurator to conduct RE, specifically for gap analysis and configuration of product specifications in a large distributed Insurance domain project in Europe. The project under study involves providing group income protection services to customers.

We created a Group Life Insurance KB using the product knowledge configurator from existing Group Life Insurance product requirement specifications. The creation of KB involves (1) identification of structural details of the product requirement specification, (2) mapping of the document structure to the product knowledge configurator model, and (3) extraction of domain knowledge from documents and its representation in product knowledge configurator. In this paper we do not discuss the details of the process of creating the KB. For details, the reader is referred to our previous work [12, 13]. Table II lists the different tasks and effort involved in creating the KB.

The details pertinent to the size and content of the KB are given in Table III. The KB has the product specifications of 25 Group Life Insurance products. Seven domain experts were involved in analysis, standardization and organization of product specifications.

TABLE II. TASKS AND EFFORT INVOLVED IN CREATING KNOWLEDGE BASE

| Task | Task Type | Effort |
|---|----------------|----------------|
| Identification of structural details of the product requirement specification documents | Manual | 4 person days |
| Mapping of the document structure to the product knowledge configurator model | Semi-automated | 6 person days |
| Extracting and organizing knowledge in product knowledge configurator | Semi-automated | 9 person day |
| Review of extracted knowledge elements | Manual | 15 person days |

The KB was validated for its correctness and completeness by the product manager. This was done by comparing the product specifications written in Microsoft word with the one populated in KB.

TABLE III. PRODUCT KNOWLEDGE ELEMENTS

| Knowledge Element | Count |
|-------------------|-------|
| Risk | 2 |
| Cover | 30 |
| Attribute Group | 150 |
| Attribute | 2750 |
| Business Rule | 961 |
| Business Process | 125 |
| Sub Process | 227 |

To evaluate the effectiveness of product knowledge configurator, we conducted an experiment to compare the manual gap analysis process routinely practiced by the project team and the one carried out using the product knowledge configurator. Six product specifications corresponding to customer-specific variants were included in the gap analysis experiment. It should be noted that one of the author (PRA) was a part of the team responsible for KB creation and manual gap analysis.

A. Traditional Gap Analysis Process

In the traditional gap analysis process, the vendor's business analyst team manually compares the configured specification (created by conducting a series of RE sessions with the customer) with the baseline product specification document to detect gaps and necessary customizations. The author followed the manual approach to perform the gap analysis.

During the experiment, the author recorded the following:

- *Total number of gaps identified by domain expert (G_{Total})*: These are the total number of gaps between the

configured product specification and the baseline product specification.

- *Total number of gaps identified by the author manually (G_{Manual}):* These are actual number of gaps identified by the author manually by comparing the configured product specification and the baseline product specification.

B. Gap Analysis Using Product Knowledge Configurator

The product specification of the six products received from the customer was configured (using the approach as illustrated in section II above). The configurator generated a gap analysis report based on each of the six configured product specification and the KB specification.

During the experiment, the author recorded the following:

- *Total number of gaps identified by product knowledge configurator ($G_{PdtConf}$):* These are the total number of gaps identified by the product knowledge configurator.
- *Total number of relevant gaps identified by product knowledge configurator ($G_{PdtConfRel}$):* These are the gaps that were identified by the product knowledge configurator and considered relevant by the domain expert.

C. Effectiveness Parameters

In order to measure the effectiveness of gap analysis process using the product knowledge configurator and compare it with the manual approach, we computed precision and recall values based on the data obtained from the experiment.

Precision: We define precision as “percentage ratio of relevant gaps identified to the total number of gaps identified between the configured specification and the KB specification”

Precision (P_M) is the percentage ratio of gaps identified by the author that were considered relevant by the domain expert (to the total number gaps identified by the author)

Precision ($P_{PdtConf}$) is the percentage ratio of gaps identified by the author using the product knowledge configurator that were considered relevant by the domain expert (to the total gaps identified by product knowledge configurator). Thus,

$$P_{PdtConf} = (G_{PdtConfRel} / P_{PdtConf}) \times 100$$

We have assumed that precision (P_M) is 100 % because all the gaps that the author identify are correct. However, the author may or may not identify all the gaps. We have verified this logical assumption with domain experts.

Recall: We define recall as “percentage ratio of gaps identified to the total actual gaps that exist between the configured specification and the KB specification”

Recall (R_M) is the percentage ratio of the gaps identified by the author to the total actual gaps that exist.

Recall ($R_{PdtConf}$) is the percentage ratio of the gaps identified by the product knowledge configurator to the total actual gaps that exist. Thus,

$$R_M = (G_{Manual} / G_{Total}) \times 100$$

$$R_{PdtConf} = (G_{PdtConfRel} / G_{Total}) \times 100$$

D. Results

In this section, we present the results of the evaluation of the effective of the configurator in terms of precision and recall percentage. Table IV lists the parameters discussed in section III C above.

Precision

We computed the average precision of the gap analysis performed using the product knowledge configurator using the following formula:

$$P_{PdtConf} = (\sum G_{PdtConfRel} / \sum G_{PdtConf}) \times 100$$

$$P_{PdtConf} = 100 \%$$

Thus, the average precision of gap analysis performed using the product knowledge configurator is 100 %

Recall

We computed the average recall of the gap analysis performed using the product knowledge configurator using the following formula:

$$R_{PdtConf} = (\sum G_{PdtConfRel} / \sum G_{Total}) \times 100$$

$$R_{PdtConf} = 92.2 \%$$

Thus, the average recall of gap analysis performed using the product knowledge configurator is 92.2 %

We further computed the average recall of the gap analysis performed manually using the following formula:

$$R_M = (\sum G_{Manual} / \sum G_{Total}) \times 100$$

$$R_M = 48.05 \%$$

Thus, the average recall of manual gap analysis is 48.05 %.

TABLE IV. GAP ANALYSIS EXPERIMENT RESULTS

| Product | G_{total} | $G_{PdtConf}$ | $G_{PdtConfRel}$ | G_{Manual} |
|---------|-------------|---------------|------------------|--------------|
| P1 | 7 | 7 | 7 | 4 |
| P2 | 14 | 12 | 12 | 8 |
| P3 | 11 | 8 | 11 | 3 |
| P4 | 21 | 18 | 18 | 11 |
| P5 | 16 | 15 | 15 | 7 |
| P6 | 8 | 6 | 8 | 4 |
| Total | 77 | 66 | 71 | 37 |

The plot in Figure 2 depicts the precision and recall value computed for gaps identified in each of the product

specification manually and using the product knowledge configurator.

E. Analysis

Average precision of the gaps identified between the configured product specification and the baseline specification when using product knowledge configurator was found to be 100 %. Average recall using the product knowledge configurator was computed at 92.2 %. Recall using manual approach was found to be only 48.05 %. As discussed earlier, we have involved the RE lead and Domain Expert in charge of the project to review the results of manual gap analysis and analysis done by the product knowledge configurator. The results indicate that there is a significant improvement in the recall percentage of gaps identified using the product knowledge configurator as opposed to the manual approach. This is because the product knowledge configurator makes the unobvious lower level product details easily visible and accessible which have a high probability of getting missed when the gaps analysis is performed manually.

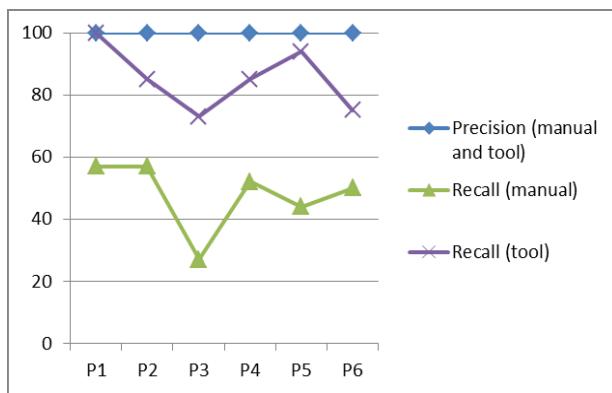


Fig. 2. Precision and Recall of gap analysis: manual and product knowledge configurator based

As an example, in Figure 3 we depict one such lower level product detail. A gap that existed in the business rules associated with Risk -> Cover -> Process -> Sub Process was omitted during the manual gap analysis due to the difficulties involved in identifying such fine grained details while manually comparing the two documents. However, when the gap analysis was done using the configurator, this gap was readily detected.

F. Threats To Validity

The following are the threats to validity:

- The recall percentage is highly dependent on the structure of a product specification desired by a customer. Each project follows its own documentation template and therefore the configurator may not be able to completely map the document structure to the configurator model. This may lead to some information loss while populating the data from the documents to the configurator. The recall percentage obtained in our experiment also indicates some amount of information loss. This dictates the need to further refine our model by testing it against different

product specification structures reflected in the templates.

- The author conducted the experiments. As the author has been involved right from conceptualization and development of the product knowledge configurator, she was aware of the functionality and capability of the tool. It remains to be seen whether the results vary when the experiment is conducted by business analysts (who are the real users of the configurator).

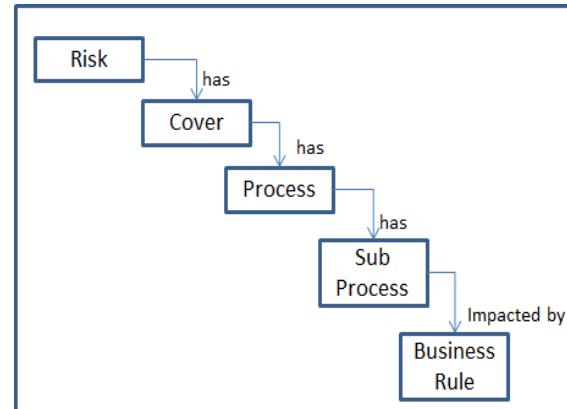


Fig. 3. An example of lower level product details

IV. DISCUSSION AND CONCLUSION

Work in the area of domain specific ontology-based frameworks for capturing and analyzing customer requirements has been reported in literature [2, 16-19]. All these works focus on either the *Manufacturing domain* or the *Production domain*. The product primitives of the Insurance domain however differ from those of the Manufacturing or the Production domain. This motivates the need for a knowledge-assisted framework for product configuration that can cater to the intricacies of Insurance domain. To the best of our knowledge, no work has been reported on ontology-based knowledge-assisted product requirements configuration in the Insurance domain.

In this paper, we present our work on a product knowledge configurator to represent product knowledge and reuse it in Requirements Engineering (RE) exercises for large projects. We capture the product primitives in the form of an ontological model. We associate various concepts in the Business Domain and Environmental Context ontologies with those in the Product Knowledge Ontology and build inference mechanisms based on this foundation. The inference mechanisms are used to provide knowledge assistance while configuring a product requirement set. Upon completing the configuration, the analyst can produce a gap analysis report from the configurator. The report highlights the 'evolution' of the product specification from its generic version to its configured variant. This information is critical to evaluate the amount of further customization that may be necessary.

We have demonstrated creation of KB and its reuse in a large Insurance project for product configuration and gap analysis. The evaluation results suggest that using the configurator, requirement analysts can perform a comprehensive gap analysis process. In contrast, analyzing

gaps manually is not only inefficient and cumbersome but also error-prone. The configurator assists business analysts in correctly identifying the gaps that exist between the configured specification and the generic product specification. This reduces the chances of redeveloping already existing features. Also, by making the tacit knowledge explicit, the configurator reduces the excessive dependence on a small number of experts.

During our interactions with the project team, we realized that our approach has the following limitations:

- The quality of KB that we create is a major determinant of success of the product knowledge configurator.
- Upfront investment in creating a KB can be a hindrance to adopting this approach. The analysts considered creating the KB as a huge overhead.
- Adoption of our approach would require a mindset change that is difficult to achieve in any organization. We observed that the analysts are so comfortable using word documents and the manual approach that they find it difficult to adopt the automated approach in spite of its benefits.
- We anticipate performance issues as the KB grows, especially due to the hierarchies that need to be traversed to detect the knowledge element instances at the leaf-level. (ref fig. 3)
- As the model is subject to refinement in the light of different product specification structures reflected in the different customer templates; the linkages between the concepts and the hierarchies could become complex. This may lead to performance issues as well.

In spite of these limitations, we are optimistic that the evaluation results would be a good motivation to receive buy-in from project teams and customer alike.

REFERENCES

- [1] Schierholt K. (2001), Process configuration: Combining the principles of product configuration and process planning. *AIEDAM*, 15, pp 411-424
- [2] Wicaksono, H.; Schubert, V.; Rogalski, S.; Ait Laydi, Y.; Ovtcharova, J., Ontology-driven requirements elicitation in product configuration systems, 4th International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2011), Montreal, Canada 2011
- [3] D. Yang, R. Miao, H. Wu, and Y. Zhou, Product Configuration Knowledge Modeling Using Ontology Web Language, *Expert Systems with Applications*, vol. 36, pp. 4399-4411, 2009
- [4] Jiao J.R., Chen C.H., Customer requirement management in product development: A review of research issues, concurrent engineering: research and applications, Vol.14, No. 3, 2006
- [5] Mesihovic S, Malmqvist J, Product data management (PDM) system support for the engineering configuration process, 14th European Conference on Artificial Intelligence ECAI 2000 Configuration Workshop August 20-25, 2000, Berlin, Germany
- [6] Pol P, Patulkar M, Method for Fit-gap Analysis in SAP ERP Projects, White Paper, Infosys.
- [7] <http://www.infosys.com/SAP/thought-leadership/Documents/methods-fit-gap-analysis.pdf>
- [8] Anish P.R., Sharma S.K., Motwani M, Ghaisas S, Knowledge – assisted product requirements configurator, PLEASE workshop, ICSE 2013, San Francisco, CA
- [9] A Falkner,I Feinerer, G Salzer, G Schenner, Solving practical configuration problems using UML, Workshop on Configuration Systems, ECAI 2008
- [10] O Djebbi, C Salinesi, Towards an automatic PL requirements configuration through constraints reasoning, Int. Workshop on Variability Modeling of Software Intensive Systems (VaMoS), Essen, Germany, January 2008
- [11] R Rabiser, D Dhungana, Integrated support for product configuration and requirements engineering in product derivation, 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007)
- [12] S. Schwarze, Specification mapping - the integration of customer requirements within a product configuration, Institute of Industrial Engineering and Management (BWI), Switzerland, Swiss Federal Institute of Technology (ETHZ), 1993
- [13] S. Ghaisas and N. Ajmeri, Knowledge-assisted requirements evolution, Accepted in Managing requirements Knowledge, (MaRK) Ed., Springer
- [14] Wikipedia: http://en.wikipedia.org/wiki/Joint_application_design last accessed on: 6th March 2014
- [15] Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M et al (2004) SWRL: a semantic web rule language combining OWL and RuleML. W3C Member Submission 21:79
- [16] Yang, D., Dong, M., Miao, R., 2008, Development of a product configuration system with an ontology-based approach *Computer-Aided Design*, 40:863-878
- [17] Kim KY, Manley DG, Yang H. Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design* 2006; 38(12):1233–50
- [18] Zhang JS, Wang QF, Wan L, Zhong YF. Product configuration modeling based on ontology. *Computer Integrated Manufacturing Systems* 2003; 9(5):344–50
- [19] Soininen T, Tiihonen J, Mannisto T, Sulonen R. Towards a general ontology of configuration. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing (AI EDAM)* 1998; 12 (4):357–72
- [20] Google search: https://www.google.co.in/?gfe_rd=ctrl&ei=pGYdU8atDYuBuATptICADQ&gws_rd=cr#q=attribute+definition Last accesses on 10th March 2014
- [21] Wikipedia: http://en.wikipedia.org/wiki/Business_Process last accessed on: 10th March 2014
- [22] D. Jannach and M. Zanker, Modeling and solving distributed configuration problems: A CSP-based approach, *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 3, pp. 603–618, 2013

Reassessing the Pattern-Based Approach for Formalizing Requirements in the Automotive Domain

Predrag Filipovikj
Mälardalen University
Västerås, Sweden
predrag.filipovikj@mdh.se

Mattias Nyberg
Scania
Södertälje, Sweden
mattias.nyberg@scania.com

Guillermo Rodriguez-Navas
Mälardalen University
Västerås, Sweden
guillermo.rodriguez-navas@mdh.se

Abstract—The importance of using formal methods and techniques for verification of requirements in the automotive industry has been greatly emphasized with the introduction of the new ISO26262 standard for road vehicles functional safety. The lack of support for formal modeling of requirements still represents an obstacle for the adoption of the formal methods in industry. This paper presents a case study that has been conducted in order to evaluate the difficulties inherent to the process of transforming the system requirements from their traditional written form into semi-formal notation. The case study focuses on a set of non-structured functional requirements for the Electrical and Electronic (E/E) systems inside heavy road vehicles, written in natural language, and reassesses the applicability of the extended Specification Pattern System (SPS) represented in a restricted English grammar. Correlating this experience with former studies, we observe that, as previously claimed, the concept of patterns is likely to be generally applicable for the automotive domain. Additionally, we have identified some potential difficulties in the transformation process, which were not reported by the previous studies and will be used as a basis for further research.

I. INTRODUCTION

New and emerging technologies are being incorporated into vehicles, leading to a significant increase of the number and complexity of the implemented functions. Currently, in industrial settings, manual inspection (peer review) is the most widespread technique for checking the correctness of the systems' specification [13]. Performing these reviews manually is becoming more difficult as the complexity of the specifications increase. Additionally, the fact that the reviewing peers may be different stakeholders, with different backgrounds and interests, and even working at different companies makes it more difficult to guarantee that the specification was properly understood and validated [6].

In these conditions, it is clear that having tool support for computer-assisted verification of the specifications would speed up the process and be beneficial. The new ISO26262 standard for functional safety of the road vehicles acknowledges this, by advocating the application of formal verification techniques at each level of system abstraction [8].

One of the main problems in order to introduce formal verification techniques in industry is the lack of formal specifications of the systems. The common practice in the

automotive industry nowadays is to specify system requirements in natural language, typically using a general purpose text editor. Requirements written like this are ambiguous, cannot be easily processed by computers and provide limited traceability; they are also most likely to be incomplete and/or inconsistent. The most desirable situation would be to have requirements expressed in some kind of logics, for instance with temporal logics such as LTL [5], CTL [5] or TCTL [1]. Requirements expressed as such can be subject to different forms of automated analysis, like consistency checks and others.

It is not realistic to believe that automotive engineers will be able to easily express system requirements using formal notations. First of all, automotive engineers are in general very knowledgeable about disciplines like mechanics, hydraulics, electronics, mechatronics, etc., but they are not so skilled in computer science and discrete mathematics. Teaching every engineer how to specify properties with temporal logics would require a great amount of time, and would be too costly to be feasible. Furthermore, it is not only the engineers who need to understand the requirements. Other stakeholders at different levels of the organization, e.g. customer service or maintenance service, must process the requirements and validate them according to their specific needs [4].

Several researchers believe that this gap can be filled with the help of software tools that will assist the engineers in the process of *automated transformation* of requirements from natural language into temporal logics [2] [9] [11]. This process is often called *formalization of requirements*. There exist some interesting tools on the market providing this functionality [7], but they are not widely accepted. More research is still needed for understanding how the automated formalization of requirements will fit into the development process of an organization.

This paper presents a preliminary study that has been conducted in order to gain further understanding of the benefits, limitations and challenges encountered when formalizing requirements in a realistic setup. This study has been performed in collaboration with Scania, one of the leading Swedish truck manufacturers. The goal of the case study is to take a

small subset of non-structured functional requirements from the E/E systems written in natural language and formalize them using the approach known as Real Time Specification Patterns, developed by Konrad and Cheng [10]. These patterns are based on the set of Specification Pattern System (SPS), initially defined by Dwyer et al. [3], but are enriched with patterns that can capture the timing aspect in the requirements. The work by Konrad and Cheng also proposed a system of restricted English grammar, through which the requirements can be expressed in specification patterns more naturally from a linguistic point of view. These patterns have been applied at least in one case study in the automotive domain [12]. A significant advantage of this approach is that each pattern maps into temporal logics, which makes the transition from the restricted English into formal notation automated. This paper summarizes the knowledge gained from this case study, with special emphasis on the challenges faced during the process, and reports future directions for research.

II. DESCRIPTION AND SETUP OF THE CASE STUDY

As already indicated, Post et al. successfully applied Real Time Specification Patterns in a case study within the automotive domain [12]. Our case study is inspired by that experience but also presents some differences. It is similar in the sense that a set of industrial non-structured requirements will be taken and converted into patterns; we will call this process *patterning* of the requirements. But it differs in the way the requirements are gathered and filtered before the patterning happens.

A. Real Time Specification Patterns

In our case study we use the Real Time Specification Pattern System (RTSP) as defined by Konrad and Cheng [10]. This set of patterns has been chosen because it provides a quantitative notion of time, which is needed for formalization of the real-time requirements. Table I shows these patterns together with their representation in restricted English grammar; the list contains 17 specification patterns. All these patterns share the characteristics of being non-recursive and prone to an automated transformation into temporal logics such as LTL, CTL and TCTL. One should notice that there is no direct mapping of *each* pattern from the RTSP into *every* temporal logics, as there can be patterns with semantics that cannot be expressed in some of the previously mentioned formal notations.

Each pattern is constituted by literal and non-literal terminals. The non-literal terminals can be either boolean expressions describing system properties, or integer values capturing timing aspects. The remaining parts of the pattern are the literal terminals, which cannot be changed. For example in the *Precedence pattern* given as: “it is always the case that if P holds, then S previously held”, P and S represent non-literal terminals and the remaining of the pattern are literal terminals. For each pattern an extent of program execution for which the requirement holds must be defined. As defined by Dwyer et al. [3], there are five scopes of program execution: *Globally* (the

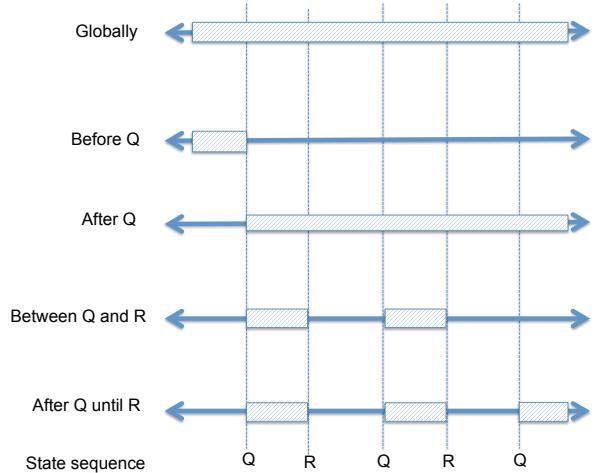


Fig. 1: Pattern scopes [12].

entire program execution), *Before Q* (until the first occurrence of the state/event Q), *After Q* (after the occurrence of the state/event Q), *Between Q and R* (any part of the program execution between the two states/events Q and R) and *After Q until R* (similar behavior as *Between* scope except that the execution continues even if the second state/event never occurs). The graphical representation of the scopes is given in Figure 1.

B. Requirements Gathering

In order to collect the requirements for the study, some engineers from Scania were contacted and asked to send requirements documents exemplifying their usual work. In response, we received four documents with a number of non-structured functional requirements written in natural language. Three of the documents were written as MS Word documents, and the other one was written in DOORS. In total, one hundred requirements were gathered. Although this number is small compared to the total number of requirements of a system, the gathered data was representative enough for the first evaluation of the patterning process.

Regarding the contents, one of the documents included requirements relative to user functions, or UFR in Scania terminology, whereas the other three contained requirements describing the behavior at the sub-system level, known as Allocation Element Requirements (AER) in Scania terminology.

After this gathering, the chosen requirements were extracted from the documents and stored in a data sheet, disposing any other information. This guarantees that only the statements marked as requirements are assessed, and not context information or other meta-data related to them. This gives an indirect measure of the quality of the requirement with respect to automated transformation: if a requirement cannot be patterned without knowing other information appearing in the document, it will most likely not be prone to an automated formalization.

A difference with respect to the previous case study is that the requirements were extracted from the documents by the researchers (not by the engineers) as they were provided,

TABLE I: Restricted English grammar patterns [10].

| | | |
|--------------------|----------------------------|--|
| Start | property | <i>scope , specification .</i> |
| Scope | scope | <i>Globally Before R After Q Between Q and R After Q until R</i> |
| General | specification | <i>qualitativeType realtimeType</i> |
| Qualitative | qualitativeType | <i>occurrenceCategory orderCategory</i> |
| | occurrenceCategory | <i>absencePattern universalityPattern existencePattern boundedExistencePattern</i> |
| | absencePattern | <i>it is never the case that P holds</i> |
| | universalityPattern | <i>it is always the case that P holds</i> |
| | existencePattern | <i>P eventually holds</i> |
| | boundedExistencePattern | <i>transitions to states in which P holds occur at most twice</i> |
| | orderCategory | <i>it is always the case that if P holds (precedencePattern precedenceChainPattern1-2 precedenceChainPattern2-1 responsePattern responseChainPattern1-2 responseChainPattern2-1 constrainedChainPattern1-2)</i> <i>, then S previously held</i> |
| | precedencePattern | <i>and is succeeded by S , then T previously held</i> |
| | precedenceChainPattern1-2 | <i>, then S previously held and was preceded by T</i> |
| | precedenceChainPattern2-1 | <i>, then S eventually holds</i> |
| | responsePattern | <i>, then S eventually holds and is succeeded by T</i> |
| | responseChainPattern1-2 | <i>and is succeeded by S , then T eventually holds after S</i> |
| | responseChainPattern2-1 | <i>, then S eventually holds and is succeeded by T , where Z does not hold between S and T</i> |
| | constrainedChainPattern1-2 | |
| Real-time | realtimeType | <i>it is always the case that (durationCategory periodicCategory realtimeOrderCategory)</i> |
| | durationCategory | <i>once P becomes satisfied, it holds for (minDurationPattern maxDurationPattern)</i> |
| | minDurationPattern | <i>at least c time unit(s)</i> |
| | maxDurationPattern | <i>less than c time unit(s)</i> |
| | periodicCategory | <i>P holds boundedRecurrencePattern</i> |
| | boundedRecurrencePattern | <i>at least every c time unit(s)</i> |
| | realtimeOrderCategory | <i>if P holds, then S holds (boundedResponsePattern boundedInvariancePattern)</i> |
| | boundedResponsePattern | <i>after at most c time unit(s)</i> |
| | boundedInvariancePattern | <i>for at least c time unit(s)</i> |

specifically without any filtering or preprocessing. In contrast, Post et al. [12] only considered behavioral requirements, and were preprocessed before the formalization. We proceed differently, because our goal is to investigate to what extent the considered specification patterns can help the verification effort. Since engineers have to verify all kinds of requirements, behavioral and non behavioral, and their starting point is typically the existing documentation, we preferred to stay as close as possible to those conditions.

C. Requirements Patterning

The patterning of the requirements was performed sequentially, one by one, by accessing only the information available in the data sheet. It was preceded by a phase in which the researchers studied the specification patterns and prepared a list containing all the patterns described in [12], which would be consulted during the process. None of the researchers involved in the process had previous experience with the specification patterns.

The patterning process consisted in the following tasks: i) identifying which pattern should be applied to the requirement and after that, ii) writing the requirement in restricted English grammar according to the chosen pattern. Since the goal of the exercise was to assess the expressiveness and adequacy of the patterns, there was no need to proceed further and obtain

the expressions in temporal logics. For the purpose of our research, we claim that a requirement is *formalizable* if there is a pattern that captures its semantics.

In some occasions, the level of ambiguity of a requirement would make it difficult to discriminate between possibly applicable patterns. In such cases, we contacted the responsible engineer and discussed the meaning of that particular requirement. This was needed for instance for determining the scope of validity of a requirement or for understanding the exact order of events. These requirements are still considered formalizable.

Some requirements could not be expressed through patterns for the reasons that will be described in Section III. Such requirements are said to be *non-formalizable*.

III. ANALYSIS OF THE RESULTS

In this section we present details about the results achieved after the patterning process.

A. Pattern Expressiveness

The results from the formalization process are presented in Figure 2. Around 70% of the requirements could be formalized with patterns; among the remaining 30% of requirements non-formalizable with patterns, there is an important group, called *phenomenon* requirements, which can still be formalized by

other means, as it will be described later on in this section. Therefore, the proportion of non-formalizable patterns is 6%.

Phenomenon Requirements: The term *phenomenon requirement* was coined by Post et al. to refer to a requirement that does not express system behavior, but gives information about data or the system configuration. These requirements cannot be mapped into a pattern, but can be expressed by means of non-literal terminals [12]. An example of Scania phenomenon requirement is:

“The signal totalFuelLevel shall receive its value from externalTotalFuelLevelIn.”

Some of the difficulties encountered in the patterning process of the phenomenon requirements is that often the scope of execution is not clearly defined. Another issue with these requirements is that sometimes they do not explicitly include sufficient data to be patterned. Let us take, for example, the requirement below, which appeared in one of the requirement documents. Note that this requirement is in fact composed by three requirements of different type, which can be extracted and considered as separate entities.

“Signal lowFuelLevelWarning shall be set to Active when input totalFuelLevel is below a predefined level. This level shall be 10% for tank size equal to or below 900 liters and 7% for tank sizes larger than 900 liters. The tank size is determined by the parameters fuelTankSizeLeft and fuelTankSizeRight.”

The statement above decomposes in the following requirements:

- 1) *“Signal lowFuelLevelWarning shall be set to Active when input totalFuelLevel is below a predefined level.”*
- 2) *“This level shall be 10% for tank size equal to or below 900 liters and 7% for tank sizes larger than 900 liters.”*
- 3) *“The tank size is determined by the parameters fuelTankSizeLeft and fuelTankSizeRight.”*

Only the first statement is a behavioral requirement, since it captures the behavior of the system after the *lowFuelLevelWarning* signal reaches some threshold (indicated as the *predefined level*). The second requirement is a phenomenon requirement used in addition to the previous requirement in order to accurately define what *predefined level* means. There is no pattern to represent this requirement, but it can be formalized using non-literal statements.

The third requirement is also a phenomenon requirement, but it cannot be formalized. It simply indicates that there is a relationship between the tank size and the parameters *fuelTankSizeLeft* and *fuelTankSizeRight*, but the exact relationship is not given. It is not possible to formalize such requirement, unless the missing relationship is defined with assistance from the engineers. Interestingly, when asked about the missing information in the requirement, the engineer answered that this information was omitted because it was considered trivial. It is well known that the existence of this “*domain knowledge*” introduces ambiguity and represents a

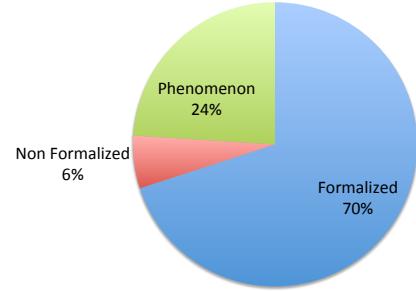


Fig. 2: Formalization results.

challenge for the specification of requirements in general. For the case of formalization of requirements, there is a need to convert this domain knowledge into formal expressions, which is also challenging.

Formalized Requirements: The second group of requirements includes the requirements that were successfully expressed with patterns. These requirements can be further divided into two big categories:

- 1) Requirements formalized without help of the engineers.
- 2) Requirements formalized with the help of the engineers.

The requirements falling into the first category typically looked like the following:

- 1) *“If lowFuelLevelWarningParam = 0, output lowFuelLevelWarning shall be set to Take No Action.”*
- 2) *“When DTC sensorShortToGround or sensorShortToBattery is set, the status of input signal fuelLevelSensor shall be set to Error”*
- 3) *“The CMS shall send a valid value in totalFuelLevel within 2 seconds from when the ECU starts.”*

In these examples, the behavior is clearly stated and the scope of the program execution can be identified, so the corresponding pattern can be found without requiring further assistance. We expressed the first two requirements with a *Response* pattern, and the third one with a *Bounded Response* pattern:

- 1) *“Globally, it is always the case that if (lowFuelLevelWarningParam = 0) holds, then (lowFuelLevelWarning = ‘Take No Action’) eventually holds.”*
- 2) *“Globally, it is always the case that if (DTC sensorShortToGround or sensorShortToBattery is set) holds, then (fuelLevelSensor = Error) eventually holds.”*
- 3) *“Globally, it is always the case that if (ECU was started) holds, then (CMS sent valid signal to totalFuelLevel) holds after at most 2 seconds.”*

However, some of the requirements that could be formalized without engineers’ assistance raised an interesting concern. It may happen that different patterns express semantics that are not easily distinguishable by a non-expert. This, combined with the inherent ambiguity of the natural language in

which the requirements are written, may result in a wrong selection of a pattern; where wrong means “not conveying the intention of the person writing the requirement”. Such cases of incorrect disambiguation of requirements are particularly interesting because they may yield the results of the associated verification activities useless. As part of a larger process, the pattern selection should be validated by other means. However, analyzing the problem of validating the correctness of the pattern selection is out of the scope of this paper, and should be addressed only in a larger empirical case study with engineers taking part in the patterning process actively. In this work, a requirement is considered formalizable if there is at least one pattern that expresses the meaning of the requirement correctly as perceived by the researchers. For instance, let us consider the following requirement:

“When parkingBrakeApplied has status ‘Error’ or ‘Not Available’ the replacement value ‘Not Set’ shall be used”

To the best of our knowledge, the semantics of the above requirement can be expressed with either *Response* pattern with *Global* scope or with an *Universality* pattern with restricted scope *After Q*. For clarification, we present the requirement expressed through both patterns, followed by their corresponding TCTL representation:

- *“Globally, it is always the case that if (parkingBrakeApplied = ‘Error’ or parkingBrakeApplied = ‘Not Available’) holds, then (parkingBrakeApplied = ‘Not Set’) eventually holds.”*

$$\text{AG}[(\text{parkingBrakeApplied} = \text{‘Error’}) \text{ OR } (\text{parkingBrakeApplied} = \text{‘Not Available’})] \rightarrow \text{AF}(\text{parkingBrakeApplied} = \text{‘Not Set’})]$$

- *“After ((parkingBrakeApplied = ‘Error’) or (parkingBrakeApplied = ‘Not Available’)), it is always the case that (parkingBrakeApplied = ‘Not Set’) holds.”*

$$\text{AG}[(\text{parkingBrakeApplied} = \text{‘Error’}) \text{ OR } (\text{parkingBrakeApplied} = \text{‘Not Available’})] \rightarrow \text{AG}(\text{parkingBrakeApplied} = \text{‘Not Set’})]$$

With the information given in the requirement alone, it is difficult to determine which of the patterns expresses the correct behavior to the full extend. The problem is that both patterns capture behaviors that are similar, but not equivalent, for instance because they are within different scopes. Potential causes of these problems are: the ambiguity of the requirement and the apparent overlapping between different patterns. The implications of choosing one pattern over the other are only evident in the following stages of the verification process, which are not considered in this case study.

Regarding the requirements for which assistance from the engineers was required, the most common difficulties were determining the scope of execution and understanding the underlying meaning. According to the Dwyer et al. [3], there are five possible execution contexts for the patterns. Interestingly, none of them captures the moment of entering a particular state

or executing a specific event. For example, let us consider the following requirement:

“Output signal lowFuelLevelWarning shall have the initial value ‘Not Active’ at start-up.”

This requirement imposes that the lowFuelLevelWarning signal shall receive some value when the system is “in” some specific state (*start-up*). None of the scopes captures the moment when the system is in a particular state but only *Before* and *After*, so we had to consult with the engineer in order to reformulate the requirement and expressed it with one of the defined scopes. For this particular requirement the *After Q* scope was applied.

Another problem identified was that sometimes concepts from different level of abstractions were used, without specifying the meaning of the high level concepts. This problem may be related to the absence of phenomenon requirements that complement the semantics of the requirement, but it needs further investigation. For example, the following requirement is apparently well specified, but contains ambiguity:

“At the shut down, the last value of the totalFuelLevel shall be stored until next start-up.”

In this requirement, concepts from different levels of abstraction are present: the *shut-down* and *start-up* states/events belong to a higher abstraction level and are not properly specified, so their meaning is ambiguous. More importantly, it is not clear what “*the last value*” in this context is, and how and where it should be stored. For such requirements patterning is impossible unless an engineer disambiguate them.

Non Formalized Requirements: The third type of requirements are the non formalized requirements. According to our experience, high complexity, high level of ambiguity and lack of information are the main reasons impeding the patterning of such requirements. We present one representative from this group:

“Signal totalFuelLevel shall be output of a filter that includes information from both FLS_vol and fuelRate to achieve a stable signal. The filter shall be implemented with a Kalman algorithm given by equations (2-4) with a feedback gain K.”

In this requirement, the relationship between the input and the output parameters has been defined with an analytical expression that is not present in the text. There is no approach that deals with transformation of analytical expressions into logic, so that the relationship between the input and the output parameters in this requirement cannot be formalized. For this reason, the requirement was classified as not formalizable.

B. Pattern Frequency

The distribution of the patterns used for formalizing the requirements in this case study is given in Figure 3. Note that the distribution includes only the requirements formalized with patterns, and not the phenomenon requirements. From a total of seventeen patterns presented by Konrad and Cheng [10], seven were used in this case study. The distribution of the

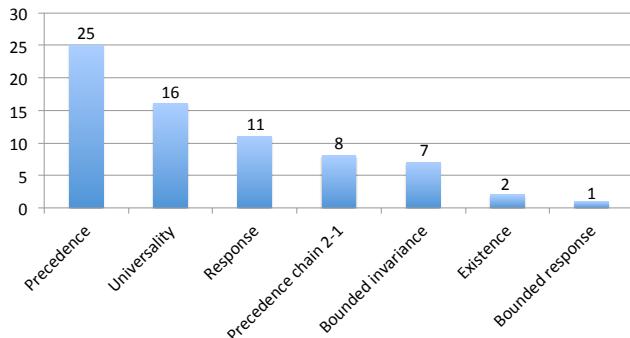


Fig. 3: Pattern frequency.

patterns is coherent with Post et al. [12] case study; this is an indication that a small number of patterns are enough to express the majority of the requirements in the automotive domain. The pattern most frequently used is *Precedence* followed by *Universality*. It is interesting is that the *Precedence chain 2-1* pattern was used in our case study, but not in Post et al's.

Another interesting difference between our results and the ones presented in the previous case study is the ratio between the *Precedence* and the *Response* patterns. In our case, most of the properties are expressed by using the *Precedence* pattern, whereas Post et al's case study uses *Response* pattern more often. These two patterns are very similar but not equivalent. This is due to the fact that the *Response* allows effects to occur without causes, while the *Precedence* allows causes to occur without effect. A number of factors can influence the application of both types of patterns, and we believe that the ambiguity of the requirements as well as the understanding of the underlying industrial systems are the most common ones. Our hypothesis is that within the verification phase, it will be possible to understand which one conveys better the requirement as originally intended by the engineer.

IV. REFLECTION ON THE EXPERIENCE

According to our observations and expectations, the patterning process resulted into a number of benefits. One is an apparent reduction of ambiguity, due to the fact that the patterns tend to capture the “*domain knowledge*” information, which is usually omitted in the requirements expressed in natural language. Additionally, in our conversations with the engineers, the patterns provided a useful support for discussing about the meaning of certain requirements. This is expected to impact positively on the communication between stakeholders, and also to improve testability of requirements.

The case study was completely focused on testing the expressiveness of the specification patterns on a functional requirements from the electrical and electronic systems installed in trucks. Driven by this specific goal, we did not take any precise measures about the effectiveness of the approach represented as patterning time per requirement. However, according to our observations, the patterning process was conducted in a

reasonable time frame, thus making it applicable in industrial settings.

The case study also revealed certain limitations of the approach. Using patterns based on restricted English grammar eliminates the need to use mathematical expressions for formalizing the requirements, but the process still requires a significant effort. We observed that as the patterning process advanced, the researchers involved in the experiment became more proficient in using the patterns and also on the understanding of the requirements, which resulted in reducing the patterning time per requirement. But finding ways of softening this learning curve is desirable.

The main challenge of the patterning process is understanding the meaning of the patterns and the scopes of execution over which the patterns hold. Post et al. [12] reported similar problems with respect to choosing the scope. Their experiment showed that the engineers favor the *Global* scope because of its simplicity. According to them, this happens because the *Global* scope captures the entire program execution, and thus it is easier to grasp. Our case study, however, showed the opposite tendency. The researchers tend to understand the patterns better and have an inclination to use as much of the defined patterns and scopes as possible. Therefore, even though the extensive use of the *Global* scope of execution can be accepted in most of the cases, finding more precise definition of the scope became a priority.

Although we believe that using restricted scopes is desirable, the implications of the effect from this decision is only visible in the next phases of the verification process. It remains as an open problem that will need further investigation.

V. CONCLUSION

This paper has discussed a case study intended to assess the suitability of using the specification patterns [10] for formalizing E/E requirements in the automotive domain. The goal was to evaluate whether such patterns provide enough expressiveness and are, at the same time, easy to apply. Even though the case study was only a preliminary study of the problem, with some inherent limitations in size, a few interesting conclusions have been drawn.

The results of applying specification patterns to our set of requirements can be considered satisfactory. Among the total set of requirements collected initially, 70% were formalized with patterns. Considering only the behavioral requirements, 92% of them were formalized. According to our observations, the patterning process reduced the ambiguity of the requirements and showed promising potential as a support to communication.

Our experience indicates that the patterns in their current form (a theoretical framework) are difficult to introduce in the industrial process. The patterns must be accompanied with support material, such as graphical representations and examples, which will make them more comprehensible for the engineers. Specifically, both the identification of patterns and the selection of scope of execution can be difficult sometimes, and may lead to wrong patterning; some extra information

is needed to avoid this problem. In addition, providing user-friendly tool support will certainly favor industrial acceptance of the methodology and is most desired.

As a conclusion, the preliminary results showed that there is a strong need for conducting a more extensive case study including more engineers from different departments and more Scania E/E systems. To further increase the relevance of the results, we are also working towards including other companies from the automotive domain into the case study. In the new setup we expect to gain deeper understanding about the patterning process of the requirements, with a particular emphasis on identifying the engineers' needs. This data will be used for providing a tool to adequately support the process with a high level of correctness. However, even with an appropriate tool support, wrong interpretations may still happen in this initial phase, and the causes should be identified and investigated. When possible, these cases should be correlated to the engineers' background. The future case study will focus on accurately measuring the time required for patterning the requirements, in order to be able to draw statistical data such as formalization time per requirement. Additional effort is also needed for identifying the right metrics to study the quality of the requirements achieved through formalization as well as measuring (qualitatively) the engineers' perception about the benefits achieved through such formalization.

ACKNOWLEDGMENT

This work was funded by the Swedish Governmental Agency for Innovation Systems (VINNOVA) under project 2013-01299.

REFERENCES

- [1] R. Alur. *Techniques for automatic verification of real-time systems*. PhD thesis, Stanford University, Stanford. 1992.
- [2] R. L. Cobleigh, G. S. Avrunin, and L. A. Clarke. *User guidance for creating precise and accessible property specifications*. In Proc. of ACM SIGSOFT Found. on Soft. Eng. (FSE). 2006. pp. 208-218.
- [3] M. B. Dwyer, G. S. Avrunin, J. C. Corbett. *Patterns in property specifications for finite-state verification*. In Proceedings of the 21st international conference on Software engineering (ICSE '99). ACM, New York, NY, USA. 1999. pp.411-420.
- [4] M. Elizabeth, C. Hull, K. Jackson, J. Dick. *Requirements Engineering, Third Edition*. Springer. 2011.
- [5] E. A. Emerson. *Temporal and modal logic*. In Handbook of Theoretical Computer Science. Amsterdam, Netherlands: Elsevier, 1995. pp. 995-1072.
- [6] N. Heumesser, F. Houdek. *Experiences in managing an automotive requirements engineering process*. In: RE, IEEE Computer Society. 2004. pp. 322-327.
- [7] H. J. Holberg, U. Brockmeyer. *ISO 26262 compliant verification of functional requirements in the model-based software development process*. White paper. Embedded World Exhibition and Conference. 2011.
- [8] International Organization for Standardization. *ISO/DIS 26262-1 - Road vehicles Functional safety*. International Organization for Standardization / Technical Committee 22 (ISO/TC 22). Geneva, Switzerland. 2009.
- [9] S. Konrad and B. H. Cheng. *Facilitating the construction of specification pattern-based properties*. In Proc. of the IEEE Int. Req. Eng. Conf. (RE). 2005. pp. 329-338.
- [10] S. Konrad, B. Cheng. *Real-time specification patterns*. In Proceedings of 27th International Conference on Software Engineering. 15-21 May 2005. pp. 372-381.
- [11] S. P. Overmyer, B. Lavoie, and O. Rambow. *Conceptual modeling through linguistic analysis using LIDA*. In Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE). 2001. pp. 401-410.
- [12] A. Post, I. Menzel, J. Hoenicke, A. Podelski. *Automotive behavioral requirements expressed in a specification pattern system: a case study at BOSCH*. In: Requirements Engineering Journal 17. March 2012. pp. 19-33.
- [13] GS. Walia, J.C. Carver. *A systematic literature review to identify and classify software requirement errors*. Inf. Softw. Technol. 2009. pp. 51(7):1087-1109.

From Architecture to Requirements: Relating Requirements and Architecture for Better Requirements Engineering

Feng Chen

Department of Computer Science and Information Systems, University of Limerick, Ireland

Lero – The Irish Software Engineering Research Centre

feng.chen@ul.ie

Abstract—The importance of software requirements is widely acknowledged. However, many software projects still exhibit inadequate Requirements Engineering (RE) practice. More importantly, dealing with Non-Functional Requirements (NFRs) remains as a challenge for software practitioners. This research aims at advancing RE practice through the co-development of requirements and architecture by utilizing the relationship between Architecturally Significant Requirements (ASRs) and Architectural Design Decisions (ADDs).

Index Terms—Architecturally Significant Requirements; Architectural Design Decisions; co-development; relationship

I. INTRODUCTION

“The hardest part of building a software system is deciding what to build ... No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.” This quote by Fredrick Brooks nearly thirty years ago has become a truism. Everybody talks how important requirements are; however, requirements practice is often neglected. This neglect is demonstrated by such facts as initial requirements are seldom more than 50% complete for 30 years [1] and about 70-85% of software rework costs are caused by requirements errors [2]. Consequently, academia has targeted research efforts at improving RE practice and has proposed various techniques and methods over the years. Yet these research contributions still haven’t visibly improved RE practice in industry. This project is another research effort that aims to improve RE practice.

To improve requirements practice, the role of architecture merits investigation. In 1994 the first international conference on requirements engineering, Garlan [5] pointed out that architectural families induce constraints on requirements in a software product line; and Jackson [5] highlighted that it is difficult to separate requirements from architecture. In 2001, Nuseibeh [19] proposed the twin-peaks model illustrating the co-development of requirements and architecture in an incremental and iterative fashion, which implies that architecture will channel constraining influences back to requirements. But the twin-peaks model provides no detail on how this co-development should be conducted. More recently, Cole [20] emphasized the changing role of architecture in that it can define requirements in a dynamic system-of-systems context. But this change is largely constrained in that particular

context and no example or empirical evidence is provided to support that change. Woods and Rozanski [4] argue that architecture can frame, constrain and inspire requirements. Such a three-way relationship from architecture to requirements extends the twin-peaks model with respect to how architecture might influence requirements. But this relationship is too general and high-level to guide the co-development of requirements and architecture. Pohl and Sikora [26] presented the COSMOD-RE (sCenario and gOal based SysteM develOpment methoD) to systematically guide the co-development of system requirements and functional architecture, which emphasizes the use of the proposed architecture solution to validate existing requirements and discover new requirements. But NFRs are not handled in this method, and it lacks a model of the relationship between requirements and architecture necessary to guide their co-development.

Requirements and architecture have a close relationship. To this author’s knowledge, research opportunities exist to improve RE practice through the co-development of architectural requirements and architecture by utilizing the relationship between them. What is the nature of this relationship and how one might leverage it to advance RE practice remains a challenge that is the focus of this research.

II. MOTIVATION AND RESEARCH QUESTIONS

Nowadays, real-world software projects are often motivated and driven by non-functional problems, such as slow processing, low scalability, high cost, and dissatisfied customer [3]. These non-functional problems highlight the NFRs, which are critical for the construction of good architecture, in order to achieve high quality software. Hence, requirements engineering and architectural design play a critical role in software development projects.

Very often, requirements elicitation mainly produces functional requirements. Capturing and dealing with NFRs remains a challenge for both academia and practitioners. The software profession has overemphasized functionality at the cost of NFRs [3]. This overemphasis succeeds in obtaining working software in the short term, but fails to achieve sustainability (ability to evolve) in the long term. Furthermore, it incurs a greater amount of technical debts which

consequently increase future maintenance costs [9]. Designing better software architecture is key in mitigating against the cost of maintenance efforts. Good architecture promotes software maintainability by exposing the dimensions where change might happen, and hence, make ramifications of change easier to understand and analyze [11].

However, software architecture is not always intentionally designed but emerges over time. One reason for this may be the way NFRs are treated. In 2003 Bass, Clements and Kazman [14] suggested that architects must be actively involved in requirements elicitation early to solicit and understand stakeholders' needs and expectations. However, in 2010, Clements and Bass [12] found that the requirements specification provides very little information needed by architects for architecting. Ameller and Franch [6] found in a survey that most NFRs are discovered solely by architects, but at the same time, architects lack of a shared understanding of NFRs. So how NFRs should be captured and treated remains a challenge, and the architect plays a central role in finding the solutions. The relevant literature [6, 28, 29] focuses on how software architects view, consider and deal with NFRs, but does not mention how they perform RE.

In software architecture, Architectural Design Decisions (ADDs) are the set of design decisions that determine and describe the resulting architecture. Jansen and Bosch [17] even regarded software architecture as a set of ADDs.

In software requirements, Architecturally Significant Requirements (ASRs) refer to those requirements that shape software architecture and drive architectural design. There is a distinction between NFRs that have an impact on architecture and those that don't. For example, some look and feel quality requirements and constraints do not shape the architecture at all, e.g. "Color coding and font size must be used to distinguish a visual element". Meanwhile, some functional requirements do drive architectural design, e.g. "The system shall provide online help", which might imply the establishment of a new component or a new connection between components.

ASRs and ADDs, although not well defined in literature to date, are the most important part of requirements and architecture respectively. Furthermore, de Boer and Vliet [7] argue that ADDs have no fundamental difference from ASRs, and it is supported by Bencomo et al [8]. This close relationship between ASRs and ADDs closes the gap between requirements and architecture. Hence, this research will focus on ASRs and ADDs when looking at the co-development of requirements and architecture.

Based on the problems identified and the knowledge obtained in the previous paragraphs, this research investigates the following questions:

RQ1. With a focus on ASRs, how do software architects perform requirements engineering?

RQ2. What approaches do software architects use to deal with the relationship between ASRs and ADDs?

RQ3. How can we utilize the result of RQ2 to better conduct requirements engineering and architectural design?

RQ1 aims to reveal patterns of the interplay between ASRs and ADDs that arises when architects do requirements

engineering, mainly when eliciting and analyzing ASRs. RQ2 necessitates a deeper dive into the interplay between ASRs and ADDs that arises when architects make ADDs with respect to ASRs, and will focus on how architecture and architectural knowledge influence requirements. How ASRs are turned into ADDs and why architects might make changes to proposed architecture solutions will be explored. Real-world problems will be identified that support the co-development of requirements and architecture. RQ3 aims to improve requirements practice by applying the findings from RQ2 to the real world.

The contribution of this research will be a conceptual framework that captures the relationship between requirements and architecture. It will address patterns that depicts how specific types of ASRs could lead to specific types of ADDs, and how specific ADDs could influence (e.g. constrain or inspire) specific ASRs. And this framework might lead to a development model that facilitates the co-development of requirements and architecture with potential time and cost saving benefits for practice.

III. STATE OF THE ART

On ASRs. ASRs constitute a large and most important part of requirements; it is the key to promote RE practice. Chen, et al [15] captured four sets of characteristics of ASRs. For example, descriptive characteristics describe ASRs as "tend to be hidden within other requirements" and "situational" etc.; indicators characteristics distinguish ASRs as "targeting tradeoffs" and "assumption breaking" etc. Eeles [13] highlighted reasons why ASRs are often overlooked, such as NFRs are less visible and familiar to most stakeholders, and proposed a conceptual architectural-requirements-questionnaire-based approach to capture ASRs. However, this approach demands extensive human-efforts (stakeholders' involvement) and is time-costly. Cleland-Huang [16] demonstrated a persona driven approach by creating architecturally significant user stories to discover ASRs and evaluate architecture solutions. However, it is hard to guarantee the completeness and accuracy of the ASRs elicited.

On ADDs. ADDs are receiving more attention and popularity now. Most researches concentrate on documenting ADDs to better preserve, manage and reuse architectural knowledge, and proposing models or approaches with regard to ADDs to support architectural design. Kruchten et al [18] described an ontology of ADDs which distinguishes four types of ADDs – existence decisions; bans or non-existence decisions; property decisions; and executive decisions – where attributes of and relationships between ADDs are also articulated.

On relating requirements and architecture. In recent years, there is a surge among researchers to focus on relating software requirements and architecture. The objectives of their researches mainly fall into three categories as follows - our research mainly falls into the last one.

- Facilitating the transition from requirements to architecture. Most research works on relating requirements and architecture fall in this category. For example, CBSP

(Component-Bus-System-Property) provides an approach to refine architectural requirements by leveraging architectural concepts so that requirements statements could be better written to exhibit clearer architectural concerns [21]. It illustrates that a carefully written ASR embodies an early ADD. Dermeval et al [27] described a model which connects requirements (in i^*) and architecture (in Acme - an Architecture Description Language [ADL]) through a ADDs metamodel using similar concepts in NFR framework. Such a model enables more precise analysis of how requirements change will influence the architecture through understanding the linkage between them – the ADDs. But this model does not address the relationship from architecture to requirements. Galster et al [22] reviewed and evaluated most transition-focused works, such as rule-based decision making and ADL, and indicated that more fundamental and further research is needed to really understand the relationship between requirements and architecture, and close their gap.

- Proposing models or approaches to promote the co-development and co-evolution of requirements and architecture. The twin-peaks model signals the concurrent and iterative development relationship between requirements and architecture [19]. However, it does not provide a detailed systematic guidance regarding to how this iterative co-development should be developed. Later, Pohl and Sikora proposed the COSMOD-RE method to support the co-development of requirements and functional architecture by providing a systematically structured process. This process addresses activities that should be taken and corresponding artifacts that should be delivered at four different abstraction levels. However, this method mainly focuses on dealing with functional requirements and does not encompass the use of existing architectural knowledge and the relationship between requirements and architecture to better engineer architectural requirements. Pimentel et al [23] utilized i^* to represent both requirements and architectural concerns in one single model so that the co-evolution of requirements and architecture can be better managed. However, it provides little explanations and insights on the relationship between requirements and architecture.
- Advancing RE through the influence from architecture. Miller et al [25] conducted a controlled study which found that students with architectural knowledge discovered more architecture-relevant requirements than those students without architectural knowledge, and are more technology-needs focused and less user-needs focused when eliciting requirements. But no empirical study from industry has supported that finding. Schwanke [24] presented an architecture-centered reference process for engineering architectural requirements. This process emphasizes the involvement of architecture concepts and description to ensure the completeness, consistency and validation of architectural requirements by reviewing requirements artifacts against those architecture artifacts. However, this process demands extensive time and human

efforts, and thereby may not be sufficiently cost-effective to be fully adopted by industries.

The literature demonstrates that architects influence requirements through their proposed architecture solutions and existing architectural knowledge. However, no detailed systematic theory is developed to elaborate upon the patterns of this influence; and no applicable and time/cost-effective guidance is proposed to advocate practitioners to better engineer requirements by using this influence. Furthermore, very few researches have focused on advancing RE practice through the co-development of requirements and architecture by utilizing the relationship between them.

IV. PROPOSED APPROACHES AND PROGRESS

This research aims to tackle problems that are not only practical (RE practice in real-world) but also theoretical (relationship between requirements and architecture). To solve these problems, proposed approaches under consideration are outlined in the following steps:

Step 1 – Conduct literature review on how to define, classify, capture and document ASRs and ADDs, and then on the relationship between requirements and architecture. This literature review helps the researcher to identify gaps and resultant research directions that can advance RE practice.

Step 2 – Select and study a software domain to facilitate an in-depth examination on the requirements and architecture of a software system, in order to obtain a deep and thorough understanding of the relationship between them.

Step 3 – Conduct case study for RQ1 and RQ2. Firstly, this case study will investigate how software architects do requirements engineering. How differently software architects perform requirements engineering compared to the way how requirements engineer do it shall be observed. It will reveal what impact the architects' architectural knowledge has on the resulting requirements. More importantly, how architects make tradeoffs on which potential requirement is to keep, refine or discard shall be explored. Comparing the ASRs and early ADDs in each iteration of the development process will contribute to an analysis of the patterns of the interplay between requirements and architecture.

Secondly, this case study will investigate the approaches that architects used to develop ASRs and ADDs. It shall identify problems in their approaches.

Step 4 – Interview architects to obtain their insights into the relationship between requirements and architecture. To fully answer RQ2, the interview data will be analyzed in combination with findings from step 3 and existing literature.

Step 5 – Construct a framework that articulates the relationship between ASRs and ADDs based on findings from step 4.

Step 6 – Synthesize a development model that facilitates the co-development of ASRs and ADDs by utilizing the framework from step 5. Such a model is designed in a way that fully complies with the framework that contains the knowledge of the relationship between ASRs and ADDs. Hence the co-development of requirements and architecture can yield

benefits through the utilization of the relationship. The model shall solve the problems identified in step 3.

Step 7 – Evaluate the framework and the model via a workshop or case study, depending on the scope and nature of the findings. The framework and the model could be applied to a real-life case study or industrial/professional participants in a workshop organized for the purpose who could be given brief tasks which require them to apply the framework/model before being asked for their feedback.

This four-year research project is now in the middle of its second year. The literature review has equipped the researcher with knowledge and directions to further this research. The domain of Contact Center Software Systems (CCSS) has been chosen and studied as the focusing domain. Three criteria guide the selection. Firstly, the kind of software system must be sophisticated and well-established so that the tacit relationship knowledge could be well developed and then embodied in the software system and in its architects' knowledge. Secondly, the selected domain should be accessible in the researcher's environment (Ireland). Lastly, the researcher should be interested in and have some basic understanding of the selected domain. Key functionalities and architecture characteristics of CCSS have been identified. Conducting an empirical case study in a CCSS company is now in progress.

V. CONCLUSIONS

Requirements and architecture have a close relationship. Requirements practice would be better conducted if not isolated from considering architectural concerns, because architecture is a source to validate, evaluate and even discover requirements. In real software projects, requirements engineering is seldom separated from architectural design as delineated by the waterfall software development model. However, the co-development of requirements and architecture is not well supported by current techniques and methods. And the relationship between requirements and architecture is far from clear and at length. This research proposes an approach to improve requirements practice, especially dealing with ASRs. It will address patterns of how architecture influences requirements and how this influence could be embedded in the co-development of requirements and architecture to improve requirements practice.

Future works include studying real projects to investigate the relationship between ASRs and ADDs, interviewing architects to reveal the characteristics of the relationship, and developing a framework and a model to embody this relationship.

ACKNOWLEDGMENT

This research is funded by the Chinese Scholarship Council and also partly supported by Lero – the Irish Software Engineering Centre. I would like to thank my supervisors Dr. Norah Power and J.J.Collins for their support in my research.

REFERENCES

- [1] C. Jones. "Positive and negative innovations in software engineering." *International Journal of Software Science and Computational Intelligence (IJSSCI)* 1.2 (2009): 20-30.
- [2] K.E. Wiegert. "Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle." Redmond: Microsoft Press. 2003.
- [3] L. Chung, and J. C. S. d. P. Leite. *On Non-Functional Requirements in Software Engineering. Conceptual Modeling: Foundations and Applications*. A. T. B. e. al., Springer: 363-379. 2009.
- [4] E. Woods, and N. Rozanski. "How software architecture can frame, constrain and inspire system requirements." *Relating Software Requirements and Architectures*. Springer Berlin Heidelberg, 2011. 333-352.
- [5] C. Shekaran, et al.. *The Role of Software Architecture in Requirements Engineering*. the First Int'l Conf. on Req. Eng. 1994.
- [6] D. Ameller, and X. Franch. "How do software architects consider Non-Functional Requirements: a survey." *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2010. 276-277.
- [7] R. C. de Boer, and H. van Vliet. "On the similarity between requirements and architecture." *Journal of Systems & Software* 82(3): 544-550. 2009.
- [8] N. Bencomo, P. Grace, and P. Sawyer. "Revisiting the relationship between software architecture and requirements: the case of dynamically adaptive systems." In: *Self-Organizing Architectures (SOAR 2009)*, September 2009, Cambridge, England
- [9] N. Brown, et al. "Managing technical debt in software-reliant systems." *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010.
- [10] M.J.C. Sousa, and H.M. Moreira. "A survey on the software maintenance process." *IEEE Proc. of Int'l Conf. on Software Maintenance*. 1998.
- [11] D. Garlan, and D.E. Perry. "Introduction to the special issue on software architecture." *IEEE Trans. Software Eng.* 21.4 (1995): 269-274.
- [12] P. Clements, and L. Bass. *Relating business goals to architecturally significant requirements for software systems*. TECHNICAL NOTE CMU/SEI-2009-TN-026, Software Engineering Institute, 2009.
- [13] P. Eeles. *Capturing Architectural Requirements*. IBM Rational Developer Works. <http://www.ibm.com/developerworks/rational/library/4706.html>. 2005.
- [14] L. Bass, P. Clements, and R. Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2003.
- [15] L. Chen, et al. (2013). "Characterizing Architecturally Significant Requirements." *IEEE Software* 30(2): 38-45.
- [16] J. Cleland-Huang, A. Czuderna, and E. Keenan. "A persona-based approach for exploring architecturally significant requirements in agile projects." *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2013. 18-33.
- [17] A. Jansen, and J. Bosch. "Software architecture as a set of architectural design decisions." In *WICSA*, pages 109–120, 2005.
- [18] P. Kruchten, P. Lago, and H.V. Vliet. "Building up and reasoning about architectural knowledge." *Quality of Software Architectures*. Springer Berlin Heidelberg, 2006. 43-58.

- [19] B. Nuseibeh. (2001). "Weaving Together Requirements and Architectures." *Computer* 34(3): 115-117.
- [20] R. Cole. "The changing role of requirements and architecture in systems engineering." *IEEE/SMC International Conference on System of Systems Engineering*, 2006.
- [21] P. Grunbacher, A. Egyed, and N. Medvidovic. "Reconciling software requirements and architectures: The cbsp approach." *Proc. of the 5th IEEE International Symposium on Req. Eng.*, 2001, 202-211.
- [22] M. Galster, A. Eberlein, and M. Moussavi. "Transition from requirements to architecture: A review and future perspective." *7th ACIS Int'l Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. SNPD 2006*, pp 9-16.
- [23] J. Pimentel, et al. "Towards Requirements and Architecture Co-evolution." *Advanced Information Systems Engineering Workshops*. Springer Berlin Heidelberg, 2012.
- [24] R.W. Schwanke. "Architectural Requirements Engineering: Theory vs. Practice." *STRAW*. 2003.
- [25] J.A. Miller, R. Ferrari, and N.H. Madhavji. "Characteristics of new requirements in the presence or absence of an existing system architecture." *17th IEEE Int'l Req. Eng. Conf.*, 2009.
- [26] K. Pohl, and E. Sikora. "The co-development of system requirements and functional architecture." *Conceptual Modelling in Information Systems Engineering*. Springer Berlin Heidelberg, 2007. 229-246.
- [27] D. Dermeval, et al. "On the use of metamodeling for relating requirements and architectural design decisions." *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013.
- [28] M. Daneva, L. Buglione, A. Herrmann. "Software Architects' Experiences of Quality Requirements: What We Know and What We Do Not Know?," *Proceedings of REFSQ*, pp. 1-17, 2013.
- [29] E. R. Poort, et al. "How architects see non-functional requirements: beware of modifiability." *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2012. 37-51.

Quantification of Social Sustainability in Software

Maryam Al Hinai

Department of Computer Science

University of Leicester

Leicester, UK

masah1@leicester.ac.uk

Abstract—Software is an essential element in the modern (largely) digitized world. Yet, social sustainability topics are, so far, under-researched in the software engineering discipline. Currently, there is neither a clear method for evaluating social sustainability of a software system at the requirements level, nor a comprehensive set of metrics for social sustainability assessment in requirements analysis. This research aims to develop a set of such metrics and an accompanying method for analyzing social sustainability requirements of software systems.

Index Terms—Software engineering, social sustainability, metrics.

I. INTRODUCTION AND MOTIVATION

As defined by the United Nations report on sustainable developments [1] “Sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs.” [1]. There are three pillars of sustainability that are commonly agreed on namely, environmental, economic and social sustainability. There are also additional (less considered) issues, such as human [2], cultural and religious. Some authors consider cultural and religious aspects as part of the social pillar [3]. The same could be said about the human aspect of sustainability.

Software is an essential element in our modern (largely) digitized world; it is playing an important part in all spheres of our daily lives - from entertainment, communication, to business. Until recently, however, the consequences of software use on sustainable development have not been considered. Having conducted the first review of sustainability-related topics in software, authors in [2] note that software sustainability is usually supported via developing models, frameworks, methods and metrics, the majority of which are currently focused on energy-efficiency, waste reduction, and cost-effective ways of software development. The social sustainability topics are, so far, under-researched in software engineering discipline.

Yet, social sustainability is an important aspect of sustainable development that is concerned with personal and communities' well-being [3-5]. The effect of software on social sustainability is ever growing, as more and more aspects of our life migrate into the digital domain (as evidenced by cybercrime, e.g. cyber stalking [6], internet fraud and cyber-pornography [7], loss of money and emotional hurt, e.g., through “online romance scam” [8], etc.). Social sustainability

has a number of traditional themes (e.g., employment) and emerging ones (e.g., empowerment) [9]. The emerging themes are found to be qualitative in nature, as opposed to the traditional quantitative ones [9]. These different themes result in different definitions (and observable indicators) tailored to fit specific domains [9]. Thus, there is a need for consensus for a set of social sustainability indicators as well as for methods to quantify social sustainability and integrate it into the everyday practice of software engineering.

The aim of this research is to support identification and quantification of social sustainability effects of software in requirements engineering in order to enable development of software systems that are conducive for social sustainability. The specific objectives of this work are to:

- 1) Produce metrics that measure the impact of software on social sustainability and
- 2) Design a method for incorporating social sustainability concerns into software requirements engineering and design (including study of how such concerns can be identified, analyzed, and modeled).

II. STATE OF THE ART

“Sustainable Software is software whose direct and indirect negative impacts on economy, society, human beings, and the environment resulting from development, deployment, and usage of the software is minimal and/or has a positive effect on sustainable development.”[10]

Social sustainability is slowly gaining grounds in computer science research [11] but much more effort is still needed in this area. In order to appraise the state of the art on topics of social sustainability and software, we have undertaken a systematic literature review (SLR) of this topic. Some of our findings are discussed below.

There are different frameworks for assessing social sustainability such as the life cycle assessment (LCA), social impact assessment and vulnerability assessment techniques.

Some of the mentioned methods were not originally developed for social concerns but were later adjusted to include social dimensions. An example is the life cycle assessment (LCA) framework which was initially used to evaluate the environmental impacts of products throughout their life cycle [12]. In [13], social dimensions were added to the LCA and used for assessing the social sustainability of biofuel industry in Indonesia. This study used such social criteria as human

rights, working conditions and cultural heritage. Another study used LCA framework and focused on the public's social acceptance of energy technologies [14]. Authors in [15] utilized the LCA model to assess food system in US. The used social indicators were associated with different stages of the food system. For example, the average age of farmers was an indicator associated with the planting and production stage [15] while the percentage of food wasted to the donated food indicator used to assess social sustainability during the end of life phase [15].

Social impact assessment refers to the analysis of the social impacts resulting from development projects [16]. In [17], the social impact assessment was part of the suggested framework for evaluating construction projects. The impacts can be related to the culture, new social classes, population changes, zoning decisions and access to public transportation [17].

Vulnerability assessment technique depends on pinpointing the most vulnerable populations and finding the social effects on them resulting from a development project [18].

Social sustainability indicators vary depending on the assessment framework used and the studied domain. Indicators are related to employment, equity, education, public services, resilience, social ties and networks, etc. Those indicators are tailored in most cases to fit a specific domain or case. The indicators will also vary depending on the level of study or granularity. For example, social sustainability of an industry can be related to social effects on the employees (internal effect) or social effects on consumers (external effects). In [19-21], indicators were derived in association with projects internal human resources and external population.

Social indicators can overlap and interchange. For instance, health can be considered an indicator on its own [22] or it can be part of public facilities indicator [23].

Software can influence a huge set of areas in social sustainability. For instance a prototype of a communication software was developed in [24] to support data sharing and communication in virtual teams and virtual organization. In order to achieve this, the authors suggested that the content should be usable and available in different contexts. The response time and scalability was measured to evaluate the achievement. In another study, software was used as a social sustainability assessment tool for biotechnological modeling [25]. In [26] software was employed as an educational game to educate students on social sustainability.

Several studies have highlighted the role of stakeholders in the process of developing or selecting suitable social sustainability indicators for their domain [13, 25, 27, 28]. This suggests the importance of involving community members, domain experts and academics in social sustainability evaluation projects.

Our SLR shows that social sustainability indicators discussed in papers discussing software (or software engineering) are much fewer than social sustainability indicators discussed for other domains. This indicates that, so far, the software engineering discipline has not adequately engaged with this topic. Thus, our research will be building the knowledge by combining software metrics knowledge and social

sustainability assessment in other domains. Here we are applying the method suggested in [29]. It was suggested that there are two approaches of achieving sustainable development. One approach is to build innovation from external knowledge to the software engineering field and the other is to create innovation from internal knowledge of the field [29].

In [11] the authors' reviewed a set of 122 papers related to sustainability in UbiCom and CHI. It is reported that 51% of the top 100 papers were derived from UbiComp/HCI community. It was also reported that social questions appeared just in twenty percent of the top 100 papers [11]. It is noticed that some of the social indicators are actually used as human computer interaction (HCI) designs evaluation. For instance, bad ergonomics has direct effect on individuals' health. As we have mentioned above, health is an indicator to social sustainability. At the same time, ergonomics was used in [30] to evaluate the usability of e-readers technology. The indicator was used but not associated to social sustainability concern.

III. RESEARCH QUESTIONS AND CHALLENGES

The research questions addressed in our work are (1) how can social sustainability requirements be identified? (2) What metrics can be used for measuring software's social sustainability? (3) What social sustainability indicators are pertinent to software?

The first question will give insights on requirements engineering and design approaches that will allow elicitation of social sustainability concerns. This will help software engineers to analyze the social sustainability aspects of software in future projects. The last two questions will provide methods for evaluating software's effects on social sustainability.

We have currently catalogued around 600 social sustainability indicators through the SLR. Reducing these (or relevant sub-set of) indicators to a reasonable set of metrics is the first challenge to be addressed. This is because we must reconcile a number of the different views on what social sustainability is and what indicators are to be used to measure social sustainability. Substantial challenge is also attributed to the qualitative nature [9] of some emerging concepts in social sustainability – these will be very difficult (if at all possible) to quantify.

Another challenge is to ensure that the method and metrics developed in this work are acceptable to the requirements engineering practitioners. Thus, the metrics and method should be light-weight yet informative. Considering the large number of indicators that we have already catalogued, producing a light-weight method for their application will not be an easy task.

Social effects are indirect and take long time to manifest [31], yet, for requirements engineering, we must be able to identify and measure social effects of software even before that software is developed. Reconciling these apparent contradictions is yet another challenge to us.

IV. PROGRESS, RESEARCH METHODS AND CONTRIBUTIONS

This research will contribute to the software and social sustainability field by:

- 1) developing a comprehensive catalogue of social sustainability concerns that originate from the systematic literature review,
- 2) building a set of metrics that support quantification of social sustainability in software requirements and design and
- 3) establishing a method to integrate social sustainability concerns into systems' requirements and design.

To the best of our knowledge, this is the first effort aiming to capture a holistic view of social sustainability and to provide a method to engineer it into software. As noted before, due to the ever increasing penetration of software in all spheres of life, it is important to understand and consider its social effects. We have already witnessed substantial change imposed through software on the traditional communication (e.g., Facebook and WhatsApp applications), shopping (e.g., eBay and Amazon), dating, learning (from YouTube videos to free online courses), political (e.g., protest and riot organization through Facebook and Twitter during Arab Spring [34] and London Riots [41]) and other practices. We are also facing such negative social consequences of software as online child bullying [36], online romance scams [8], and radicalization of youth [35]. By providing a method and metrics for engineering social sustainability into software, this work will facilitate the anticipation and prevention of such undesired consequences [37].

To accomplish the above contributions, the following research will be carried out (table 1):

Stage 1: conduct systematic literature review. The aim here is to discover the common social sustainability indicators, social sustainability metrics, the role of software in social sustainability and the indicators of software's social sustainability. A set of about 600 indicators has already been catalogued through SLR. These indicators are to be aggregated into coherent groups. The grouping will be done through light-weight grounded theory analysis (which is to say that grounded theory coding will be applied to extracts from documents, rather than their whole texts).

Stage 2: explore software metrics used in requirements engineering and design, study and how they are built and used. This is to be done through additional literature review as well as by extracting information on software sustainability metrics from stage 1.

Stage 3: develop a set of metrics that support quantification of social sustainability in software requirements and design. This will be achieved through application of the software engineering metric construction techniques (obtained from stage 2) to the set of social sustainability indicators (obtained from stage 1). The work at stages 1 and 2 has no particular restrictions on data collection and study domain. However, we foresee that each indicator group developed in stage 1 will be substantial enough to define a domain (e.g., we can provisionally identify such groups as security, equality, etc.). In stage 3 each indicator group will be considered a separate domain. Thus, at this stage, domain-specific stakeholders will

also be consulted, as their knowledge and perspectives on relevance of a particular metric are essential. We expect to conduct focus groups and workshops with such stakeholders on per-indicator-group basis.

Stage 4: construct a method to incorporate social sustainability concerns into systems' requirements engineering and design. This stage should address 2 distinct issues:

a) generalizing the metric construction method worked on stages 1-3, and

b) integration of metrics developed in stages 1-3 into the framework of requirements engineering and design methodologies in current practice (i.e., primarily agile development).

The generalization of the metric development work into a method is necessary as – due to the expected substantial number of indicator groups - it will not be possible to work thought all the groups within this PhD project. Yet, having provided the complete (currently available through SLR, stage 1), set of social sustainability indicators and a validated method for metric construction for a given indicator group, we will enable further metric development for the remaining indicator groups.

Integration of developed social sustainability metrics into a software engineering methodology is also necessary if these metrics are to be used by others. Here such issues as social sustainability concern identification, modeling, and treatment would be considered. Identification, for instance, could be potentially supported through a natural language processing (NLP) tool to capture social sustainability indicators from software requirements document [32]. Another potential method is the use of software patterns to provide guidelines for elicitation of social sustainability requirements [33]. We do not intend to provide any new concepts or techniques here, but to demonstrate how the existing ones can account for the social sustainability concerns.

Stage 5: conduct case studies on several e-government initiatives in Oman to evaluate the effectiveness of the metrics. The case study approach will be used to evaluate the completed work, however, we are also aware of a number of validity concerns related to the case study methodology. One such concern is, for instance, related to the selection of representative cases reflecting larger population. This is due to the in-depth investigation that will lead to limited number of cases that makes results generalization an issue [38]. According to [39], this can be resolved by analytic generalization (theoretical generalization) and theory testing. Analytic generalization is “the extraction of a more abstract level of ideas from a set of case study findings – ideas that nevertheless can pertain to newer situations other than the case(s) in the original case study. For case study evaluations, the analytic generalization should aim to apply to other concrete situations and not just to contribute to abstract theory building.” [38] This can be achieved by relating the findings to literature and finding overlaps and gaps as well as by conducting additional case studies to reproduce findings [38].

TABLE I. RESEARCH STAGES AND METHODS

| Stage | Task | Method |
|-------|--|--|
| 1 | Collect social sustainability indicators | Systematic literature review |
| 2 | Discover software metrics | Reading and results from stage 1 |
| 3 | Develop set of metrics | Integrating results from stage 1 and 2 |
| 4 | Construct software method | Use natural language processing tool and software patters. |
| 5 | Evaluate the metrics | Case study (e-government) |

Currently, we are in the first stage of the project. A systematic literature review is in progress. We have taken advantage of the digital libraries to conduct our search. We focused on computer science and engineering digital libraries and decided to use ACM digital library, IEEE, Scopus and Springer link. We have also used Web of science and ASSIA to derive the social concern of our research. The search terms used were social sustainability and metrics, social sustainability and indicators and software and social sustainability. Studies were selected based on their relevance to the defined research questions. Some primary results have informed the work outlined in this paper. A further paper on the systematic literature review is currently in submission [40].

V. CONCLUSION

As discussed above, sustainable development is slowly taking a center stage in research. Software engineers are also increasingly focusing on sustainable development issues for software applications [4].

We are currently developing a catalogue of software's social sustainability indicators, which will lead to development of a set of social sustainability metrics and evaluation method for social sustainability in software requirements. We are facing a number of challenges in our project, including the task of folding numerous indicators into a manageable set of metrics, estimating/measuring long-term effects of software which is not even developed yet, and integrating social issues into the normally functionality- (and quality) focused engineering discipline. We are looking forward to discussions and feedback on avenues of potential progress with these (and possibly other, unnoticed by us) challenges.

ACKNOWLEDGMENT

I would like communicate my appreciation to my supervisor, Dr. Ruzanna Chitchyan, for her continuous guidance and support during this research.

REFERENCES

- [1] Brundtland, G., Our common future: The world commission on environment and development, 1987, Oxford: Oxford University Press.
- [2] Penzenstadler, B., et al., Sustainability in software engineering: a systematic literature review. 2012.
- [3] McKenzie, S., Adult and Vocational Education for Social Sustainability: A New Concept for TVET for Sustainable Development, in Work, Learning and Sustainable Development, J. Fien, R. Maclean, and M.-G. Park, Editors. 2009, Springer Netherlands. p. 177-186.
- [4] Penzenstadler, B. and H. Femmer, A generic model for sustainability with process- and product-specific instances, in Proceedings of the 2013 workshop on Green in/by software engineering2013, ACM: Fukuoka, Japan. p. 3-8.
- [5] Willis, P., S. McKenzie, and R. Harris, Introduction: Challenges in Adult and Vocational Education for Social Sustainability, in Rethinking Work and Learning, P. Willis, S. McKenzie, and R. Harris, Editors. 2009, Springer Netherlands. p. 1-9.
- [6] Stephenson, P. and R. Walter, Cyber Crime Assessment. 2012: p. 5404-5413.
- [7] Chung, W., et al., Fighting cybercrime: a review and the Taiwan experience. Decision Support Systems, 2006. 41(3): p. 669-682.
- [8] Whitty, M.T. and T. Buchanan The Online Romance Scam: A Serious Cybercrime. 2012. 15, 181-183 DOI: 10.1089/cyber.2011.0352.
- [9] Colantonio, A., Social sustainability: linking research to policy and practice. 2009.
- [10] Dick, M., S. Naumann, and N. Kuhn, A Model and Selected Instances of Green and Sustainable Software, in What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience, J. Berleur, M. Hercheuil, and L. Hilty, Editors. 2010, Springer Berlin Heidelberg. p. 248-259.
- [11] Knowles, B., et al., Exploring sustainability research in computing. 2013: p. 305.
- [12] Handbook of Life Cycle Assessment: Operation Guide to ISO Standards, J.B. Guinee, Editor 2002, Kluwer Academic Publishers: Secaucus, NJ, USA.
- [13] Manik, Y., J. Leahy, and A. Halog, Social life cycle assessment of palm oil biodiesel: a case study in Jambi Province of Indonesia. The International Journal of Life Cycle Assessment, 2013. 18(7): p. 1386-1392.
- [14] Assefa, G. and B. Frostell, Social sustainability and social acceptance in technology assessment: A case study of energy technologies. Technology in Society, 2007. 29(1): p. 63-78.
- [15] Heller, M.C. and G.A. Keoleian, Assessing the sustainability of the US food system: a life cycle perspective. Agricultural Systems, 2003. 76(3): p. 1007-1041.
- [16] Porter, A.L. and F.A. Rossini, Technology Assessment/Environmental Impact Assessment: Toward Integrated Impact Assessment. Systems, Man and Cybernetics, IEEE Transactions on, 1980. 10(8): p. 417-424.
- [17] Valdes-Vasquez, R. and L.E. Klotz, Social Sustainability Considerations during Planning and Design: Framework of Processes for Construction Projects. Journal of Construction Engineering & Management, 2013. 139(1): p. 80-89.
- [18] Pearsall, H., From brown to green? Assessing social vulnerability to environmental gentrification in New York City. Environment and Planning C: Government and Policy, 2010. 28(5): p. 872-886.
- [19] Labuschagne, C., A.C. Brent, and S.J. Claassen, Environmental and social impact considerations for sustainable project life cycle management in the process

- industry. *Corporate Social Responsibility and Environmental Management*, 2005. 12(1): p. 38-54.
- [20] Sarkis, J., M.M. Helms, and A.A. Hervani, Reverse logistics and social sustainability. *Corporate Social Responsibility and Environmental Management*, 2010. 17(6): p. 337-354.
- [21] Brent, A. and C. Labuschagne, Social Indicators for Sustainable Project and Technology Life Cycle Management in the Process Industry (13 pp + 4). *The International Journal of Life Cycle Assessment*, 2006. 11(1): p. 3-15.
- [22] Halme, M., C. Jasch, and M. Scharp, Sustainable homeservices? Toward household services that enhance ecological, social and economic sustainability. *Ecological Economics*, 2004. 51(1-2): p. 125-138.
- [23] Landorf, C., Evaluating social sustainability in historic urban environments. *International Journal of Heritage Studies*, 2011. 17(5): p. 463-477.
- [24] Fiumara, G., et al., Knowledge representation in virtual teams: a perspective approach for synthetic worlds, in collaborative networks for a sustainable world, L. Camarinha-Matos, X. Boucher, and H. Afsarmanesh, Editors. 2010, Springer Berlin Heidelberg. p. 619-625.
- [25] von Geibler, J., et al., Accounting for the social dimension of sustainability: experiences from the biotechnology industry. *Business Strategy and the Environment*, 2006. 15(5): p. 334-346.
- [26] Gennett, Z.A., J.A. Isaacs, and T.P. Seager. Developing a social capital metric for use in an educational computer game. in *Sustainable Systems and Technology (ISSST)*, 2010 IEEE International Symposium on. 2010.
- [27] Meul, M., et al., MOTIFS: a monitoring tool for integrated farm sustainability. *Agronomy for Sustainable Development*, 2008. 28(2): p. 321-332.
- [28] Saadatian, O., K.B. Sopian, and E. Salleh, Adaptation of sustainability community indicators for Malaysian campuses as small cities. *Sustainable Cities and Society*, 2013. 6: p. 40-50.
- [29] Yu, G. and Y. XiaoQiu. The Two Approaches to Sustainable Development of the theory of Software Process Models. in *Information Management, Innovation Management and Industrial Engineering (ICIII)*, 2010 International Conference on. 2010.
- [30] Pearson, J., G. Buchanan, and H. Thimbleby, HCI design principles for ereaders, in *Proceedings of the third workshop on Research advances in large digital book repositories and complementary media2010*, ACM. p. 15-24.
- [31] Naumann, S., et al., The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, 2011. 1(4): p. 294-304.
- [32] Sampaio, A., et al., EA-Miner: Towards Automation in Aspect-Oriented Requirements Engineering, in *Transactions on Aspect-Oriented Software Development III*, A. Rashid and M. Aksit, Editors. 2007, Springer Berlin Heidelberg. p. 4-39.
- [33] Roher, K. and D. Richardson. Sustainability requirement patterns. in *Requirements Patterns (RePa)*, 2013 IEEE Third International Workshop on. 2013.
- [34] Bruns, A., T. Highfield, and J. Burgess, The Arab Spring and Social Media Audiences: English and Arabic Twitter Users and Their Networks. *American Behavioral Scientist*, 2013. 57(7): p. 871-898.
- [35] Rashid, A., et al., Who Am I? Analyzing Digital Personas in Cybercrime Investigations. *Computer*, 2013. 46(4): p. 54-61.
- [36] Rybnicek, M., R. Poisel, and S. Tjoa. Facebook Watchdog: A Research Agenda for Detecting Online Grooming and Bullying Activities. in *Systems, Man, and Cybernetics (SMC)*, 2013 IEEE International Conference on. 2013.
- [37] Penzenstadler, B., et al., Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century. *Software*, IEEE, 2014. 31(3): p. 40-47.
- [38] Yin, R.K., Validity and generalization in future case study evaluations. *Evaluation*, 2013. 19(3): p. 321-332.
- [39] Bryman, A., *Social research methods*. 4th ed. 2012, Oxford: Oxford University Press. xli, 766 p.
- [40] Al Hinai, M. and R. Chitchyan, Social Sustainability Indicators for Software: Initial Review. *Third International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy) to be held at RE 2014. In submission*.
- [41] Bohannon, J., Tweeting the London Riots. *Science*, 2012. 336(6083): p. 831.

Improving Collaborative and Post-WIMP Systems through Requirements Specification

Miguel A. Teruel

LoUISE Research Group, Computer Systems Department
University of Castilla – La Mancha
Albacete, Spain
miguel@dsi.uclm.es

Abstract—A proper requirements specification is paramount for achieving the quality of the developed software products. However, well-known Requirements Engineering (RE) techniques lack of enough expressiveness to model the requirements of CSCW systems (*Computer Supported Collaborative Work*). This is due to the inherent complexity of collaboration among users and their need of awareness. Moreover, the way in which users interact with CSCW systems have evolved greatly to more sophisticated interfaces, beyond the classical desktop computer environments, to those called Post-WIMP (Windows, Icons, Menus, Pointer). Awareness is magnified in such a way that users have to be aware of their context: the artifacts with which to interact, his/her own capabilities as well as those of the others. All this awareness is necessary to allow them to collaborate in virtual and/or augmented environments. This PhD thesis aims at solving this problem by developing a RE framework able to deal with the requirements of CSCW and Post-WIMP systems, making emphasis on the awareness requirements about user's context.

Index Terms—Requirements, quality, CSCW, Post-WIMP, awareness, CSRM.

I. MOTIVATION

A CSCW system is a kind of software product enabling groups of users to get involved in a common task or goal. Thereby, by means of these CSCW systems, people are able to communicate with each other as well as to work in a collaborative manner. This implies that the group of users can coordinate their activities, solve problems, edit documents and diagrams, or negotiate by using specific technologies. Moreover, the way in which user interacts with computer systems has evolved towards a new paradigm beyond the classical windows, icons, menus and pointing devices (WIMP). This new paradigm, namely Post-WIMP, is based on virtual reality, gesture recognition and wearable computers [30] (among other technologies) that makes the development of these systems significantly different than that based on classical WIMP interaction.

Accordingly, one of the main challenges in the development of avant-garde interactive systems, especially when considering their end-users, is the stage corresponding to the requirements specification. The LoUISE (*Laboratory of User Interfaces & Software Engineering*) Research Group has been carrying out research activities related to the development of both WIMP and Post-WIMP systems for the last 15 years. In this sense, this PhD thesis aims at following the same research line, thus improving

the development of collaborative and Post-WIMP systems from a quality perspective, focusing on the specification requirements.

II. SOCIO-ECONOMICAL STAKES & INDUSTRY ISSUES

Because of the popularity gained by the Internet in the last years, lots of classical working scenarios have changed to Web-based collaboration systems. Nowadays, millions of users share information and efforts by remotely collaborating on multiple application domains, either in a synchronous or asynchronous way. In this manner, the expected results of this PhD thesis are focused on several application domains: administration, business, education, medicine and so on.

Moreover, the introduction of new social and collaborative features in most of the existing applications makes necessary to develop avant-garde tools, environments and methodologies that facilitate its support. Concretely, collaboration tends to be performed in a more natural way by taking advantage of Post-WIMP interaction. Consequently, the limitations of current interactive systems developments need to be overcome, especially to provide support to the *context awareness*.

III. RESEARCH QUESTIONS

Taking into account both a literature review and the previous research carried out by the LoUISE group, the following shortcomings and needs for the development of Collaborative & Post-WIMP systems were identified:

1. We have identified different proposals for the development of CSCW systems [10, 28, 35] and Post-WIMP systems [18, 20, 29]. Nevertheless, they only focus on design activities hardly dealing with the first stages of the development process itself.
2. For the development of CSCW systems, a critical issue is the identification of the user's *awareness* [11–13], that is, the identification of artifacts to manipulate, users to interact with, etc. Such awareness concept is mainly associated to characteristics of the software product itself and accordingly, to requirements awareness.
3. Another important issue when developing CSCW & post-WIMP systems is related to *adaptation*, especially when a system has to be used in different contexts as claim international standards, such as ISO/IEC 9126-1:2001 [17] or ISO/IEC 25010:2011 [15].

Taking into account the previous shortcomings and needs, our research questions (RQ) were established as follows:

- *RQ1*: What, if any, are the shortcomings of current RE techniques for specifying CSCW systems? If so, how can they be improved?
- *RQ2*: What, if any, are the shortcomings of current RE techniques for specifying Post-WIMP systems? If so, how can they be improved?
- *RQ3*: What awareness requirements can improve the user-perceived quality of CSCW & Post-WIMP systems?
- *RQ4*: What adaptation requirements would help to make the CSCW & Post-WIMP systems self-adaptable to their context?

With these RQs, the main technical challenges associated with them are mainly related to the novelty of the studied domain. On the one hand, CSCW systems have been widely dealt from the nineties. However, almost none effort have been put into their RE specification. On the other hand, avant-garde Post-WIMP applications are relatively new-fangled systems. Consequently, we will have to deal with a not-very-mature family of emerging systems for which RE is just starting.

IV. STATE OF THE ART

As aforementioned, a CSCW system is a software product where several users work in a collaborative manner, thus performing collaboration, communication and coordination tasks (3C) [5]. In order to perform such tasks, users must be aware of who is able to collaborate, where they are working, what they are doing, when they did certain action and so on. These and several other elements of which the user of a CSCW system must be aware of were identified in the Gutwin's *Workspace Awareness* (WA) [11].

The problem arises when trying to specify the requirements of these CSCW systems by using well-known RE techniques such as Use Cases [19], Viewpoints [8] or Goal-Oriented (GO) [21]. That is because these techniques are not able to properly specify CSCW requirements due to the complexity of the 3C tasks and the user's awareness needs. Because of that, well-known RE techniques should be adapted in order to deal with CSCW requirements properly. Moreover, awareness in Post-WIMP applications goes beyond the collaborators themselves, thus creating the need of being aware of the whole context that surrounds the user [9]. As an example, in classical WIMP application, it could be relatively easy to know where the artifacts we are working with are (e.g. the paragraphs in a document or the items in a 2D classical game). However, when interacting with Post-WIMP systems, users must be aware of the system's artifacts by using several senses. For instance, in a Virtual Reality (VR) environment, a user cannot see something that is behind him/her although it can be perceived by using the rear speakers of a surround sound system, as shown in Fig. 1. Because of this need of context awareness, the set of awareness requirements needs to be enriched to properly specify the requirements of Post-WIMP applications.

Finally, although there are several proposals for the development of Post-WIMP systems, they are mainly focused on the design and implementation of their user-interface, not

paying special attention to early stages of such developments, i.e. the requirements specification [10, 28, 35]. Because of this lack of RE techniques for Post-WIMP systems, its requirements are usually specified by using well-known techniques, in a similar way to CSCW systems. The limitation here is that well-known RE techniques focus on WIMP systems, so that the introduction of the context adaptation during the software development is just delayed to the latest stages of the process. This means that, basically, the WIMP systems are able to self-adapt to the device screen and/or communication capabilities [27, 34]. Nevertheless, context adaptation for Post-WIMP applications is far more complex due to the richest real-time hardware-depending adaptation. For instance, in VR games, a user could be aware of receiving damage by means of vibrators [24][26] located on several parts of his/her body. However, if these wearable vibrators were unavailable for any reason, it would be needed an adaptation to provide the user with a visual method such as to show him/her a body representation in a VR display device highlighting the damaged area. Because of these complex adaptation features, it is necessary to deal with them from the very early stage of the development, i.e., since the requirements specification. As far as we know, some works related to adaptation requirements have been carried out [1][2], although none of them focus on Post-WIMP.



Fig. 1. User playing a Post-WIMP First Person Shooter

V. RESEARCH METHOD

In order to perform this PhD thesis, the *design science research* method [14] is being applied. Hence, in order to be compliant to this method, its seven research guidelines will be followed:

1. *Design as an Artifact*: The aim of this research is to produce a series of artifacts: a CSCW & Post-WIMP RE specification language (model), a methodology to apply it (method) and a CASE tool (construct).
2. *Problem Relevance*: The application domain of this PhD thesis is CSCW and Post-WIMP systems. The former is getting more and more popular since the nineties. The latter constitutes the foundation of future (and even present) applications. Because of these reasons, the problem relevance is ensured.
3. *Design Evaluation*: Our research will be led by empirical evaluations. In this sense, all the developed models, methods and constructs will be evaluated by means of

- controlled experiments and case studies in order to assess and improve them.
4. *Research Contributions*: As anticipated in the first guideline, the clear contribution of this research is a framework for the specification of CSCW and Post-WIMP systems including a modeling language, a supporting tool, a set of methodological guidelines, and a collection of user's awareness requirements.
 5. *Research Rigor*: Throughout the research process, several Software Engineering methods [23, 55], statistical techniques [4, 54] and international standards [15–17] will be used in both the construction and evaluation of the artifacts.
 6. *Design as a Search*: The research will follow an iterative process whose goal is the quality improvement of CSCW and Post-WIMP systems by providing proper RE techniques. Then, the set of obtained results will be iteratively evaluated and improved until the previous goal is fulfilled.
 7. *Communication of Research*: The research result must be presented to a technologically-oriented audience. With this aim, there will be detailed publications related to the research artifacts, focused on RE practitioners, enabling them to use the research artifacts. Moreover, it will also provide researchers with the required documentation for further extension and evaluation of the research artifacts.

Within this PhD thesis, the PhD candidate and the LoUISE group members will participate in the performance of the next activities, whose timing is depicted in Fig. 2:

- *State of the art*: Research issues related to RE, CSCW, Post-WIMP environments, awareness interpretations and adaptation will be identified. This activity will be performed repeatedly throughout the whole PhD thesis.
 - *RE for CSCW Systems*: Development of a RE integrated proposal for the specification of CSCW systems, supported by Model-Driven Development (MDD) techniques taking advantage of previous experiences of the LoUISE group [32][33]. Moreover, modifications of processes described by existing CSCW methodologies will be also considered.
 - *RE for Post-WIMP Systems*: The suitability of the developed CSCW proposal will be assessed for Post-WIMP systems. Consequently, those modifications required to achieve its suitability will be analyzed.
 - *Awareness Requirements*: Identification, development and assessment of an integrated awareness interpretation gathering the most important awareness requirements that a user participating in Post-WIMP environments (whether collaborative or not) can need or expect.
 - *Adaptation Features and RE*: Analysis of the RE proposal with regard to the adaptation features necessary for post-WIMP systems. Moreover, the proposal will be integrated into a new version of UsiXML (*USer Interface eXtensible Markup Language*) [7, 22]. This language, in whose definition the LoUISE group has been actively involved, is widely accepted by the community for expressive capabilities for specifying adaptation needs of WIMP systems. Therefore, it will be extended or adapted for dealing with Post-WIMP systems.
- *Support*: Throughout this PhD thesis, several tools will be implemented in order to put into practice the developed proposal.
 - *Validation*: In order to validate the developed proposal, several prototypes will be implemented focusing on different application domains such as e-learning or rehabilitation or Post-WIMP CSCW in which the LoUISE group has previous experience [6, 25, 31].
 - *Diffusion*: Throughout the whole process of this PhD thesis, different publications will be submitted to both national and international conferences and journals in order to receive feedback. Moreover, as a part of this diffusion process, the dissertation associated to this PhD will be written.

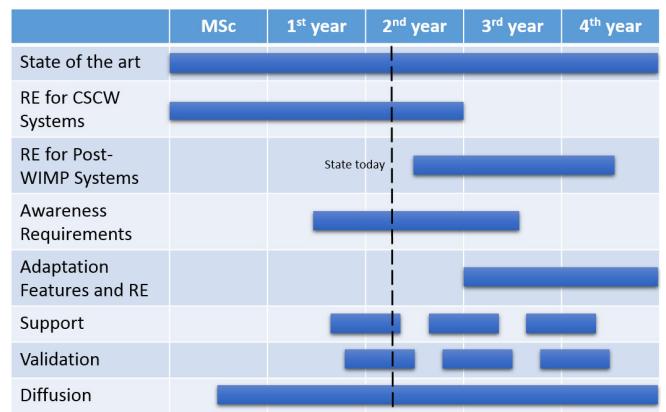


Fig. 2. Schedule of activities

VI. CONTRIBUTIONS

In this Section, the contributions of this PhD thesis so far are presented. This will be done by describing the contributions according to the RQ they are related to in different subsections. Finally, the list of the papers published by the candidate will be presented.

A. RQ1: RE for CSCW Systems

The very first task performed when this PhD thesis started was the study and comparison of the existing RE techniques when dealing with CSCW requirements. With this aim, Use Cases [19], Viewpoints [8] and GO [21] were empirically evaluated, concluding that GO approaches were the most adequate to specify CSCW requirements [41]. This results led us to analyze the most widely adopted GO proposals, NFR Framework [3], KAOS [21], *i** [56], in order to identify which one was the most suitable for CSCW systems. After a second empirical evaluation [37, 45], *i** was selected as the best alternative. However, it was also detected that *i** suffers a lack of expressiveness to specify features of CSCW requirements, such as to specify the collaboration among users as well as the need of awareness to perform such collaboration. These shortcomings led us to the development of CSRML (*Collaborative Systems Requirements Modeling Language*), a GO approach based on *i** to properly specify the requirements of complex CSCW system, able to deal with 3C tasks and awareness needs [42, 49]. Moreover, CSRML was empirically validated with regard to *i** by means of a family of experiments

[44, 51], showing that it is more understandable than i^* when CSCW systems have to be specified.

Once CSRML was evaluated, a CASE tool, namely CSRML Tool 2012 (CT'12), was developed in order to help RE practitioners to specify CSCW requirements with CSRML. Hence, CT'12 was created as a Visual Studio extension to model and validate CSRML requirements specifications in a visual and straightforward manner [48]. Moreover, the usability of the tool was empirically evaluated by means of a controlled experiment [38] that provided really positive results. Thanks to this evaluation, several usability flaws were detected and solved in the following release of the tool CT'13.

Furthermore, the aforementioned CSRML language and its supporting tool were combined to define CSRMF (*Collaborative Systems Requirements Modeling Framework*) [46]. This framework also includes a set of guidelines helping analysts to identify and model the whole set of requirements of collaborative systems. Hence, by using the CSRMF guidelines, a CSCW system can be specified starting from the identifications of its (groups of) users and roles, and finishing with the collaboration, awareness and quality factors specification.

Finally, CSRML has proven to be suitable for different application domains, as demonstrated on its brand new extension called CSRML4BI [53]. By means of this extension, CSRML has been used to model collaboration in systems of Business Intelligence (BI) by specifying a clear view of the involved tasks, their participants, and the information to be shared between them.

B. RQ2: RE for Post-WIMP Systems

Regarding the second RQ, its analysis has been started recently so that, there are no significant results yet. Concretely, the first work is being carried out and consisted in specifying the requirements of a complex highly-awareness-demanding and context-depending Post-WIMP system by using the CSRML language. The main goal of this work is to assess whether CSRML provides enough expressive power for specifying post-WIMP systems. It is worth noting that the case study is being defined with the support of several researchers of the LoUISE group who are experts in Virtual Reality (VR).

C. RQ3: Awareness Requirements

As far as this RQ is concerned, the state of the art regarding awareness interpretations has been already carried out. The first conclusion has been that Workspace Awareness (WA), as defined by Gutwin et al. [11], is the most widely and applied interpretation when talking about CSCW. Nevertheless, when it is applied to Post-WIMP applications, such as Augmented Reality (AR) or VR environments, there are some awareness requirements that WA cannot identify. For instance, awareness of future facts or social characteristics cannot be identified by using WA. Because of that, a deeper analysis of the available awareness interpretations has been carried out paying special attention to those concepts shared by more than one interpretation.

The result of this analysis was an awareness requirements catalogue able to represent the most frequent concepts. This catalogue led us to develop a new awareness interpretation for Post-WIMP systems, able to represent the awareness requirements of such systems, that includes present, past, future and social features [36]. Furthermore, the proposed awareness interpretation was empirically evaluated in order to determine whether all the considered awareness elements would improve the user experience of Post-WIMP systems. Concretely, two of the most widely played contemporary 3D games were chosen to perform this evaluation. It was concluded that all the elements of this interpretation improve the user's gaming experience, especially, those which were not already identified in WA.

Finally, based on WA, a design pattern has been defined to provide CSCW designers with the scaffolding for their collaborative applications [40]. This pattern includes elements that feature CSCW systems implementing WA, such as collaborative actions among users, shared artifacts and views, common goals, etc.

D. RQ4: Adaptation Features and RE

The last one of the RQs is in an early stage yet. So far, we are working on the integration of UsiXML and CSRML. UsiXML, the specification language for user interface design, is being adapted or extended to support Post-WIMP applications. Hence, CSRML is intended to be the part of the next UsiXML version by providing proper traceability between requirements and the following stages of the software development process. However, since this integration is being carried out, there are no publications related to this RQ.

E. Publications

The papers published by the candidate up to date related to his PhD Thesis are shown in Table I.

VII. CONCLUSIONS AND PENDING WORK

This PhD thesis, which started with the study of the suitability of current RE techniques for CSCW systems, has achieved its greatest accomplishment with CSRML, a Goal-Oriented RE language to model CSCW system requirements. It is worth noting that our research process has been fully led by empirical evaluations, thus assessing every result. Nevertheless, in spite of the suitable acceptation that CSRML has had (as the number of downloads of both the tool and the different papers shows), a considerably amount of work remains to be done.

Regarding RQ1, the CSRMF framework has been defined. However, this framework requires a full methodology in order to help RE practitioners to elicitate, model, analyze, verify and validate CSCW systems. With this aim, the already available methodologies for i^* are being studied for their integration into the CSRMF framework.

The work related to the RQ2 has just started. In this sense, once the case study is specified, the suitability of the current CSRMF framework for Post-WIMP requirements will be assessed. It is very likely that this assessment will lead to an adaptation of CSRMF so that it provides the necessary guidance for specifying the requirements of Post-WIMP systems.

TABLE I. PUBLICATIONS RELATED TO THIS PDH THESIS TO DATE

| Publication | RQ |
|--|-----|
| <i>JCR Journals</i> | |
| Analyzing the Understandability of Requirements Engineering Languages for CSCW Systems: A Family of Experiments [51] (IST, Q1) | 1 |
| A CSCW Requirements Engineering CASE Tool: Development and Usability Evaluation [38] (IST, Q1) | 1 |
| An Awareness Interpretation for Contemporary Computer Games (submitted, HCI, Q2) [36] | 3 |
| CSRML: A Framework for Modeling Requirements of CSCW Systems (submitted, RE, Q2) [46] | 1 |
| <i>International CORE Conferences</i> | |
| An Empirical Evaluation of Requirement Engineering Techniques for Collaborative Systems [41] (EASE, CORE A) | 1 |
| A Comparative of Goal-Oriented Approaches to Modelling Requirements for Collaborative Systems [37] (ENASE, CORE B) | 1 |
| CSRML: A Goal-Oriented Approach to Model Requirements for Collaborative Systems [49] (ER, CORE A) | 1 |
| A Design Pattern for Representing Workspace Awareness [40] (CSCWD, CORE B) | 3 |
| CSRML4BI: A Goal-Oriented Requirements Approach for Collaborative Business Intelligence [53] (ER, CORE A) | 1 |
| <i>International Workshops</i> | |
| An extension of i^* to Model Requirements for CSCW Systems Applied to Conference Preparation System with Collaborative Reviews [42] | 1 |
| Assesing the Understandability of Collaborative Systems Requirements Notations: an Empirical Study [44] | 1 |
| CSRML Tool: a Visual Studio Extension for modeling CSCW Requirements [48] | 1 |
| <i>Book Chapters</i> | |
| Comparing Goal-Oriented Approaches to Model Requirements for Collaborative Systems [45] (selected paper from ENASE) | 1 |
| <i>National Conferences ("Published in Spanish")</i> | |
| Modeling Collaborative Systems Requirements with CSRML ^a [50] | 1 |
| CSRML Tool: A Tool for Modeling Collaborative Systems Requirements ^a [47] | 1 |
| Analyzing the Understandability of Requirements Engineering Languages for CSCW Systems: A Family of Experiments (disclosure of relevant published papers) [43] | 1 |
| A CSCW Requirements Engineering CASE Tool: Development and Usability Evaluation (disclosure of relevant published papers) [39] | 1 |
| Multidevice Tool for Collaborative Editing EMF-based Models ^a [52] | 2,3 |

Regarding RQ3, a new awareness interpretation has been defined to gather the awareness requirements of CSCW & Post-WIMP systems. This interpretation has been positively evaluated by means of a survey filled in by users. However, in order to evaluate it more exhaustively, further experiments will be carried out involving in a more active way to the subjects. Moreover, and also related to RQ1, the WA design pattern purposed will be extended to consider this new awareness interpretation. Once the new pattern be developed, and taking advantage of the MDD approach, the necessary transformations will be specified to allow the designers to instantiate this pattern from the requirements, specified by means of CSRML, in an automatic way. This support is considered a must for this proposal to provide the necessary traceability throughout the whole development process.

Finally, since the analysis of the RQ4 has just started, there is a great deal of work to do. The first task is the integration of CSRML in the new version of UsiXML. This integration is being defined to provide CSRML with expressive power for specifying adaptability requirements necessary for Post-WIMP systems. Currently, it has been detected that the expressiveness

for tasks, already provided by CSRML, should be extended or adapted to consider the *context* of the system (i.e. available hardware, communication channels, user capabilities and even environmental factors).

ACKNOWLEDGMENTS

This work is funded by the Spanish Ministry of Economy and Competitiveness and by the FEDER funds of the EU under the project Grant insPIre (TIN2012-34003). It has also been funded by Spanish Ministry of Education, Culture and Sport thanks to the FPU scholarship (AP2010-0259).

REFERENCES

- [1] N. Bencomo, "Requirements for Self-adaptation," in: Lämmel, R. et al. Eds. Generative and Transformational Techniques in Software Engineering IV. pp. 271–296, Springer Berlin Heidelberg (2011).
- [2] G. Brown, B.H.C. Cheng, H. Goldsby, J. Zhang, "Goal-oriented specification of adaptation requirements engineering in adaptive systems," International workshop on Self-adaptation and self-managing systems (SEAMS'06). p. 23, ACM Press, Shanghai, China (2006).
- [3] L.M. Cysneiros, E. Yu, "Non-Functional Requirements Elicitation," in: do Prado Leite, J.C.S. and Doorn, J.H. Eds. Perspectives on Software Requirements. pp. 115–138, Springer US (2004).
- [4] O. Dieste, E. Fernández, R. García Martínez, N. Juristo, "Comparative analysis of meta-analysis methods: when to use which?," 15th International Conference on Evaluation & Assessment in Software Engineering (EASE'11). pp. 36–45, IET, Durham, UK (2011).
- [5] C.A. Ellis, S.J. Gibbs, G. Rein, "Groupware: some issues and experiences," Commun. ACM 34 (1), 39–58 (1991).
- [6] H. Fardoun, F. Montero, V. López-Jaquero, "eLearnXML: Towards a model-based approach for the development of e-Learning systems considering quality," Adv. Eng. Softw. 40 (12), 1297–1305 (2009).
- [7] J. Figueroa-Martínez, V. López-Jaquero, F.L. Gutiérrez Vela, P. González, "Enriching UsiXML language to support awareness requirements," Sci. Comput. Program. 78 (11), 2259–2267 (2013).
- [8] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, M. Goedicke, "Viewpoints: A Framework for Integrating Multiple Perspectives in System Development," Int. J. Softw. Eng. Knowl. Eng. 2 (1), 31–57 (1992).
- [9] A.S. Garcia, J.P. Molina, D. Martínez, P. González, "Enhancing collaborative manipulation through the use of feedback and awareness in CVEs," 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI'08). p. 1, ACM Press, Fusionopolis, Singapore (2008).
- [10] J.L. Garrido, P. Paderewski, M.L. Rodríguez-Almendros, M.J. Hornos, M. Noguera, "A Software Architecture Intended to Design High Quality Groupware Applications," Software Engineering Research and Practice. pp. 59–65, (2005).
- [11] C. Gutwin, S. Greenberg, "A Descriptive Framework of Workspace Awareness for Real-Time Groupware," Comput. Support. Coop. Work 11 (3), 411–446 (2002).
- [12] C. Gutwin, S. Greenberg, "Effects of awareness support on groupware usability," SIGCHI Conference on Human Factors in Computing Systems. pp. 511–518, ACM Press, Los Angeles, USA (1998).
- [13] C. Gutwin, S. Greenberg, "The effects of workspace awareness support on the usability of real-time distributed groupware," ACM Trans. Comput. Interact. 6 (3), 243–281 (1999).
- [14] A.R. Hevner, S.T. March, J. Park, S. Ram, "Design Science in Information Systems Research," MIS Q. 28 (1), 75–105 (2004).
- [15] ISO/IEC 25010, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, (2011).
- [16] ISO/IEC 25062, Software engineering—Software product Quality Requirements and Evaluation (SQuaRE)—Common Industry Format (CIF) for usability test reports, (2006).
- [17] ISO/IEC 9126-1, Software engineering - Product quality: Quality model, (2001).
- [18] R.J.K. Jacob, A. Girouard, L.M. Hirshfield, M.S. Horn, O. Shaer, E.T. Solovey, J. Zigelbaum, "Reality-based interaction: a framework for post-

- WIMP interfaces," SIGCHI Conference on Human Factors in Computing Systems (CHI'08). pp. 201–210, (2008).
- [19] I. Jacobson, "Use cases - Yesterday, today, and tomorrow," *Softw. Syst. Model.* 3 (3), 210–220 (2004).
- [20] H.-C. Jetter, M. Zöllner, J. Gerken, H. Reiterer, "Design and Implementation of Post-WIMP Distributed User Interfaces with ZOIL," *Int. J. Hum. Comput. Interact.* 28 (11), 737–747 (2012).
- [21] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," 5th IEEE International Symposium on Requirements Engineering (RE'01). pp. 249–262, IEEE Comput. Soc, Washington DC, USA (2001).
- [22] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, V. López-Jaquero, "USIXML: A Language Supporting Multi-path Development of User Interfaces," in: Bastide, R. et al. Eds. *Engineering Human Computer Interaction and Interactive Systems.* pp. 200–220, Springer Berlin Heidelberg (2004).
- [23] S. Linkman, B.A. Kitchenham, D. Law, "DESMET: a methodology for evaluating software engineering methods and tools," *Comput. Control Eng. J.* 8 (3), 120–126 (1997).
- [24] D. Martinez, S. Kieffer, J. Martinez, J.P. Molina, B. Macq, P. Gonzalez, "Usability evaluation of virtual reality interaction techniques for positioning and manoeuvring in reduced, manipulation-oriented environments," *Vis. Comput.* 26 (6-8), 619–628 (2010).
- [25] D. Martínez, J.-Y.L. Lawson, J.P. Molina, A.S. García, P. González, J. Vanderdonckt, B. Macq, "A Framework to Develop VR Interaction Techniques Based on OpenInterface and AFreeCA," in: Campos, P. et al. Eds. 13th International Conference on Human-Computer Interaction (INTERACT'11). pp. 1–18, Springer Berlin Heidelberg, Lisbon, Portugal (2011).
- [26] J. Martínez, A.S. García, M. Oliver, J.P. Molina, P. González, "VITAKI: A Vibrotactile Prototyping Toolkit for Virtual Reality and Videogames," *Int. J. Hum. Comput. Interact.* (2014).
- [27] I. Mohamed, J.C. Cai, S. Chavoshi, E. de Lara, "Context-aware interactive content adaptation," 4th international conference on Mobile systems, applications and services (MobiSys'06). p. 42, ACM Press, Uppsala, Sweden (2006).
- [28] A.I. Molina, M.A. Redondo, M. Ortega, U. Hoppe, "CIAM: A methodology for the development of groupware user interfaces," *J. Univers. Comput. Sci.* 14 (9), 1435–1446 (2008).
- [29] J.P. Molina, A Structured Approach to the Development of 3D User Interfaces. University of Castilla-La Mancha (2008).
- [30] J.P. Molina, A.S. García, D. Martinez, F.J. Manjavacas, V. Blasco, V. López, P. González, "The development of glove-based interfaces with the TRES-D methodology," *Proc. ACM Symp. Virtual Real. Softw. Technol. - VRST '06* 216 (2006).
- [31] F. Montero, V. López-Jaquero, E. Navarro, E. Sánchez, "Computer-aided relearning activity patterns for people with acquired brain injury," *Comput. Educ.* 57 (1), 1149–1159 (2011).
- [32] F. Montero, E. Navarro, "ATRIUM: Software Architecture Driven by Requirements," 14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2009). pp. 230–239, IEEE, Potsdam, Germany (2009).
- [33] E. Navarro, J.A. Mocholi, P. Letelier, I. Ramos, "A metamodeling approach for requirements specification," *J. Comput. Inf. Syst.* 46 67–77 (2006).
- [34] E.G. Nilsson, J. Floch, S. Hallsteinsen, E. Stav, "Model-based user interface adaptation," *Comput. Graph.* 30 (5), 692–701 (2006).
- [35] V.M.R. Penichet, M.D. Lozano, J.A. Gallud, R. Tesoriero, "User interface analysis for groupware applications in the TOUCHE process model," *Adv. Eng. Softw.* 40 (12), 1212–1222 (2009).
- [36] M.A. Teruel, E. Navarro, V. Lopez-Jaquero, F. Montero, P. González, "An Awareness Interpretation for Contemporary Computer Games (submitted)," *Human–Computer Interact.* (2014).
- [37] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "A Comparative of Goal-Oriented Approaches to Modelling Requirements for Collaborative Systems," in: Maciaszek, L.A. and Zhang, K. Eds. 6th International Conference on Evaluation of Novel Software Approaches to Software Engineering (ENASE'11). pp. 131–142, SciTePress, Beijing, China (2011).
- [38] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "A CSCW Requirements Engineering CASE Tool: Development and Usability Evaluation," *Inf. Softw. Technol.* 56 (8), 922–949 (2014).
- [39] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "A CSCW Requirements Engineering CASE Tool: Development and Usability Evaluation (submitted)," XIX Jornadas de Ingeniería del Software y Bases de Datos (JISBD'14). , Cádiz, Spain (2014).
- [40] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "A Design Pattern for Representing Workspace Awareness," 18th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD'14). , Hsinchu, Taiwan (2014).
- [41] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "An empirical evaluation of requirement engineering techniques for collaborative systems," 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE'11). pp. 114–123, IET, Durham, UK (2011).
- [42] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "An extension of i* to Model Requirements for CSCW Systems Applied to Conference Preparation System with Collaborative Reviews," 5th International i* Workshop (iStar'11). pp. 84–89, , Trento, Italy (2011).
- [43] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "Analyzing the Understandability of Requirements Engineering Languages for CSCW Systems: A Family of Experiments (on press)," XVIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'13). , Madrid (Spain) (2013).
- [44] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "Assessing the Understandability of Collaborative Systems Requirements Notations: an Empirical Study," 1st International Workshop on Empirical Requirements Engineering (EmpiRE'11). pp. 85–92, , Trento, Italy (2011).
- [45] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "Comparing Goal-Oriented Approaches to Model Requirements for CSCW," in: Maciaszek, L.A. and Zhang, K. Eds. *Evaluation of Novel Approaches to Software Engineering.* pp. 169–184, Springer-Verlag, Berlin/Heidelberg, Germany (2012).
- [46] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "CSRML: A Framework for Modeling Requirements of CSCW Systems (submitted)," *Requir. Eng.* (2014).
- [47] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "CSRML Tool: una Herramienta para el Modelado de Requisitos de Sistemas Colaborativos," XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'12). , Almería (Spain) (2012).
- [48] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "CSRML Tool: a Visual Studio Extension for Modeling CSCW Requirements," 6th International i* Workshop (iStar'13). pp. 122–124, CEUR Vol-978, Valencia (Spain) (2013).
- [49] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "CSRML: A Goal-Oriented Approach to Model Requirements for Collaborative Systems," in: Jeusfeld, M. et al. Eds. 30th International Conference on Conceptual Modeling (ER'11). pp. 33–46, Springer Berlin Heidelberg, Brussels, Belgium (2011).
- [50] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, P. González, "Modelado de Requisitos de Sistemas Colaborativos con CSRML," XVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD'11). pp. 639–652, , A Coruña, Spain (2011).
- [51] M.A. Teruel, E. Navarro, V. López-Jaquero, F. Montero, J. Jaen, P. González, "Analyzing the Understandability of Requirements Engineering Languages for CSCW Systems: A Family of Experiments," *Inf. Softw. Technol.* 54 (11), 1215–1228 (2012).
- [52] M.A. Teruel, A.C. Rodriguez, E. Navarro, P. González, "Herramienta Colaborativa Multidispositivo para la Edición de Modelos basada en EMF (submitted)," XIX Jornadas de Ingeniería del Software y Bases de Datos (JISBD'14). , Cádiz, Spain (2014).
- [53] M.A. Teruel, R. Tardío, E. Navarro, A. Maté, P. González, J. Trujillo, R. Terol Muñoz, "CSRML4BI: A Goal-Oriented Requirements Approach for Collaborative Business Intelligence," 33rd International Conference on Conceptual Modeling (ER'14). , Atlanta, USA (2014).
- [54] B.J. Winer, D.R. Brown, K.M. Michels, *Statistical Principles in Experimental Design*, McGraw-Hill Humanities/Social Sciences/Languages (1991).
- [55] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, *Experimentation in Software Engineering*, Springer Berlin Heidelberg, Berlin, Heidelberg (2012).
- [56] E.S.K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," 3rd IEEE International Symposium on Requirements Engineering (ISRE'97). pp. 226–235, IEEE Comput. Soc. Press, Annapolis, USA (1997)

Stakeholders' Social Interaction in Requirements Engineering of Open Source Software

Tanmay Bhowmik

Department of Computer Science and Engineering

Mississippi State University

Mississippi State, USA

tb394@msstate.edu

Abstract—Requirements engineering (RE) involves human-centric activities that require interaction among different stakeholders. Traditionally, RE has been considered as a centralized, collocated, and phase-specific process. However, in open-source software (OSS) development environment, the core RE activities are iterative and dynamic and follow a rather decentralized software engineering paradigm. This crosscutting characteristic of open-source RE can be conceptualized using the “Twin Peaks” model that weaves RE together with software architecture. Although many weaving mechanisms have been proposed in recent years, lack of theoretical underpinning limits a mechanism’s applicability and usefulness in different scenarios. In this research proposal, we hypothesize stakeholders’ social interaction as an ecologically valid weaving mechanism of the “Twin Peaks” in open-source RE. We further outline a concrete research plan to examine the generalizability of this weaving mechanism for three activities: requirements identification, requirements implementation, and creativity in RE. Carrying out this research plan will enable us to gain valuable insights to generate guidelines for enhancing software engineering practice in relevant areas.

Index Terms—Requirements engineering; twin peaks; open-source RE; requirements identification; requirements implementation; creativity in RE

I. MOTIVATION

Requirements engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software system, and the contexts in which it will be used [1]. The RE process includes the elicitation, modeling, analysis, negotiation, prioritization, and realization of the requirements that the intended software shall meet [1]. Traditionally, RE has been considered as a centralized, collocated, and phase-specific process associated with individual projects or project components. Much of the traditional RE has focused on models and techniques in order to aid identification and documentation of stakeholders and their needs in a form that can be analyzed, communicated, agreed upon, and eventually realized and validated. However, in the open-source software (OSS) development environment, RE activities need to focus on generation, negotiation, adaptation, realization, and maintenance of requirements in a decentralized, iterative, and dynamic software-intensive ecosystem [2]. RE activities in the OSS development paradigm are therefore no longer part of a centralized process specific to a particular phase of software development. Rather these activities are spread, tangled, or

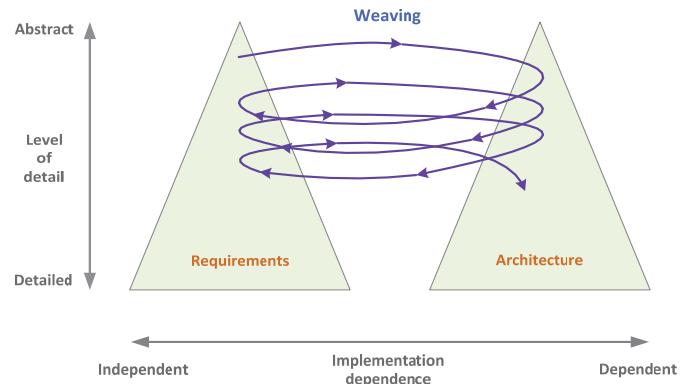


Fig. 1. The “Twin Peaks” of requirements and architecture (adapted from [3]).

otherwise intertwined with the overall software development process in a decentralized manner.

Recent research has highlighted the crosscutting characteristics of RE in relation to other software engineering activities. In particular, Nuseibeh [3] identified the highly intertwining nature of RE and software architecture and characterized this relationship in the “Twin Peaks” model. This model, shown in Figure 1, emphasizes on the equal status and an iterative co-development of requirements and architecture. The distinction between requirements and architecture has its root in Jackson’s seminal works in RE [4], [5], [6]. Specifically, Jackson distinguished the problem domain and the machine domain. Requirements are located in the problem domain, transformed via the shared phenomena between the two domains, and eventually realized by the source code executing in the machine domain. Despite the distinction, the “Twin Peaks” model emphasizes the weaving mechanism that connects the problem and machine domains. Many approaches have been proposed to instantiate the weaving, including model-based test-driven development [7], architecture and hardware constraints [8], and engineering architecturally significant requirements [9], [10]. However, existing methods mostly follow the traditional, centralized software engineering paradigm. As a result, little is known about how to weave the “Twin Peaks” in the decentralized environment of open-source RE.

Research on the nature of RE in OSS development has emerged only recently. Alspaugh and Scacchi studied the

RE activities of OSS systems and emphasized that many successful OSS projects do not follow the classical, one-time RE process [11]. Dissimilarities between traditional RE and open-source RE was also stressed by Xiao *et al.* [12]. A seminal paper was presented by Ernst and Murphy [13] exploring that in open-source RE stakeholders often capture requirements less formally and elaborate the requirements only after the implementation begins. They introduced the notion of “just-in-time” requirements to characterize the tightly coupled stakeholder relationship in open-source development. Their study shows that a crucial factor in the success of OSS projects is the social interaction among the stakeholders from both problem and machine domains [4], [5]. It is through stakeholders’ social interaction that technical aspects are clarified and organizational dependencies are resolved [14]. Inspired by the work of Ernst and Murphy [13], we formulate the central hypothesis of our research.

Central hypothesis: In decentralized OSS development, stakeholders’ social interaction serves a unified weaving mechanism of the Twin Peaks, which offers insights into a wide range of RE activities.

This research proposal introduces an ecologically valid [15] weaving mechanism of the “Twin Peaks” that promises a further understanding of RE in a decentralized environment. Successful execution of this research work may also help us generalize the weaving mechanism in the closed-source world where stakeholders’ social interaction is often dominant. As several means of stakeholders’ social interaction exist in the software world, the biggest challenge in our research is associated with operationalizing social interaction manageable in the scope of this work. Thus, this research abstract describes the state-of-the-art from which we derive the central hypothesis and operationalize stakeholders’ social interaction in Section II. We present a concrete plan to test our hypothesis in Section III. Section IV discusses the current progress and contributions, followed by Section V providing the summary and future work of this research.

II. STATE OF THE ART

In this section, we describe the background and bedrocks of our research. The goal is to lay the foundation for a novel and improved RE framework for OSS development that will be introduced in Section III. To that end, we first present Jackson’s conceptualization and distinction of problem and machine domains [4], [5]. Then we explain the “Twin Peaks” model of requirements and architecture [3], followed by a synopsis on the OSS RE [2] and stakeholders’ social interaction in OSS development.

A. The Problem-Machine Environment

Software development is an engineering process with software being the end product. This product shall strive to meet the stakeholder’s goals, needs, and desires, which are commonly referred to as requirements in software engineering. In one of the foundational papers, Jackson teased out the meaning of requirements by distinguishing two domains:

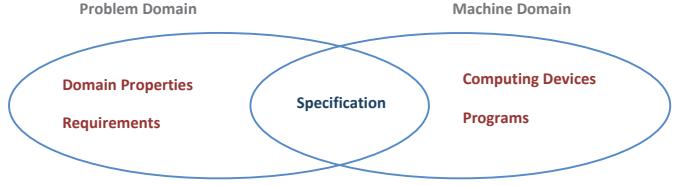


Fig. 2. The problem-machine environment (adapted from [4]).

the problem domain and the machine domain [4]. Figure 2 shows this conceptualization. The problem domain consists of properties and phenomena of the application that give rise to the stakeholder’s requirements, whereas the machine domain is private to the intended software and the computing devices in which the software operates.

Requirements are primarily located in the problem domain, as shown in Figure 2. This has several important implications to RE. First, the elicitation and identification of requirements shall focus on the real-world needs of users, customers, and other stakeholders. Second, the transformation of requirements to a software-intensive solution is very open-ended and creative in nature because computing technologies provide many opportunities for possible realization of the requirements. Third, the validation of requirements cannot be performed only in the machine domain (e.g., through verification of specification); rather, the implemented software-intensive system is valid only if it corresponds to the fulfillment of stakeholder’s goals. As shown in Figure 2, “specification” serves as the bridge between the two domains. Jackson used designation and assertion to define specification in formal logic (e.g., first-order predicate logic, temporal logic, and deontic logic) [4]. Meanwhile, he emphasized that RE is not a branch of pure mathematics or logic: the meaning of requirements depends crucially on the interpretation by the stakeholders in the two domains.

B. The Twin Peaks Model

Building upon Jackson’s conceptualization, Nuseibeh proposed the “Twin Peaks” model as a way of showing the intertwining relationship between RE and software architecture [3]. Figure 1 illustrates this model. The key observation here is that requirements can influence the architecture that designers select or develop, whereas candidate architectures can restrict designers from meeting particular requirements [3]. The model emphasizes the equal status of requirements and architecture and places each in a peak. The vertical axis indicates the level of details from abstract to detailed, while the horizontal axis presents implementation dependence. The model concurrently starts the requirements and architectural specification at an abstract level considering requirements specification as the problem structure and architectural specification as the solution structure. According to the model, the abstract requirements and architecture go through a simultaneous iterative process that progressively produces separate requirements and architectural specifications at a more detailed level.

It is interesting to note that the “Twin Peaks” model can be viewed as a generalization of Jackson’s conceptualization in that “weaving” is used as a general mechanism to link together the problem and machine domains, and “specification” is an instantiation of weaving in Figure 2. In fact, several methods and techniques have been proposed in recent years to implement “weaving”, or in other words, to traverse the “Twin Peaks” [16]. Mou and Ratiu [7] introduced model-based test-driven development as a mechanism for linking formalized requirements and architectures in embedded software development. Loft *et al.* suggested that hardware platform plays a crucial part in the requirements and architectural development of software systems with hardware constraints. Engineering architecturally significant requirements [9], [10] could be another means of traversing the “Twin Peaks” as they oftentimes influence a software system’s design decisions [17]. However, these existing solutions are rather isolated and mostly built upon traditional centralized software engineering paradigm. As a result, the success in one situation may not be applicable to other scenarios or activities. In this research proposal, we aim to create a theoretically sound weaving mechanism that can be applied to a wide range of “Twin Peaks” activities, especially in a modern open-source RE context.

C. OSS RE and Stakeholders’ Social Interaction

Only recently researchers have begun to understand RE in OSS development. In his seminal work, Scacchi [18] identified a set of twenty-odd different types of what he called ‘software informalisms’ in use across a wide variety of OSS projects. According to Scacchi [18], informalisms like instant messaging and internet relay chat (IRC) provide socially lightweight mechanisms for communicating and coordinating project knowledge. In a further study, Scacchi *et al.* [19] examined OSS systems from five different application domains and found that very often a new requirement in an OSS system is informally captured through a story telling or a user experience at the initial stage. Ernst and Murphy [13] advanced our understanding about OSS RE by studying the just-in-time requirements management where requirements identification and realization are tightly coupled. In just-in-time RE certain requirements are informally captured and later clarified through stakeholders’ social interaction during implementation.

In order to ensure sustainability in the modern competitive market, OSS systems are also subject to deliver new and improved features to the users realizing creative new requirements in short release cycles. Social interaction has been found to be influential on productivity and new idea generation. For example, Pirolli [15], [20] suggested that the way stakeholders interact with each other could influence the novelty of their ideas. Several means of such interactions among stakeholders have been identified in the literature. For example, comments or activities in issue tracking systems, email, and IRC are widely considered as means of social interaction among different stakeholders [21], [22], [23]. Research has also found a dominant use of issue tracking systems by the developers and

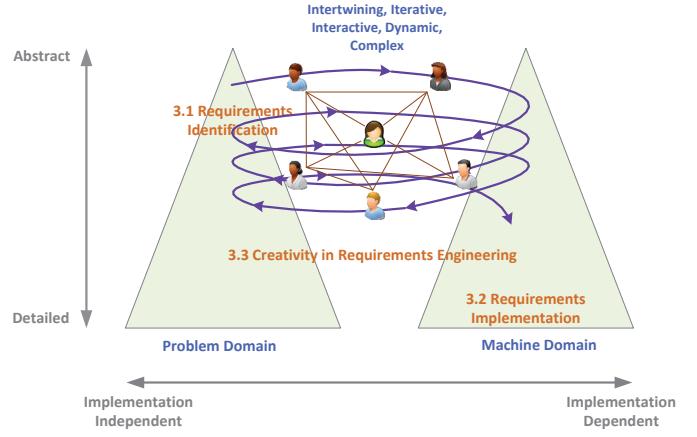


Fig. 3. Using stakeholders’ social interaction to unify the “Twin Peaks” in OSS development.

other stakeholders to interact among themselves and to keep the requirements on track [13].

Building upon the research on OSS RE, we try to leverage stakeholders’ social interaction to tackle the weaving of “Twin Peaks”. This weaving mechanism indeed incorporates the real world scenarios, thereby can address a wide range of evolving and practical “Twin Peaks” activities in a decentralized software development paradigm. Following the literature, we also plan to consider stakeholders’ activities over the issue tracking systems in order to operationalize their social interaction. Next, we tease out our research plan to test stakeholders’ social interaction as a weaving mechanism of the “Twin Peaks”.

III. RESEARCH PLAN

The central hypothesis of our work is that: “stakeholders’ social interaction serves a unified weaving mechanism to link together the Twin Peaks, which offers insights into a wide range of open-source RE activities”. Figure 3 conceptualizes our proposed research. We hypothesize that such a mechanism has two main advantages: (1) it is general to account for a wide spectrum of RE activities; and (2) it is ecologically valid [15] so that the research findings can provide potentially transformative benefits to software practitioners.

To test the central hypothesis, we choose to study three activities, namely requirements identification, requirements implementation, and creativity in RE as shown in Figure 3. Horizontally, these activities span the problem and machine domains and their intersection. Vertically, they also cover different levels of details in the engineering process. Our choice follows purposive sampling in that the findings from these representative cases can be used for theoretical generalization [24]. Although, the chosen activities are apparently discrete, we plan to study them from a common perspective, i.e., stakeholders’ social interaction. Furthermore, in our study design, these three activities not only strike a proper balance over the “Twin Peaks”, but also represent typical and critical cases so that the lessons learned can be informative to a wide variety of important RE tasks. Next we describe in more detail

about each activity and outline our research plan along these dimensions.

A. Requirements Identification

Requirements identification, a human-centric activity involving multiple stakeholders, includes determining the organizational or customer needs that must be addressed by the delivered artifacts [25]. It has been long accepted that stakeholders' social interaction is indispensable in successful identification and negotiation of requirements [26], [27], [19]. In fact, tools and techniques, such as IBM Jazz, focus groups, and repertory grids, have been developed to support social interaction in such RE activities [26], [28], [29], [30]. Despite the development of such technological supports, much less is known about how stakeholders' social interaction and their organizational arrangements influence requirements identification.

Study Plan: Following the conceptual framework of our research in Figure 3, we propose to investigate *how stakeholders' social interaction contributes to new requirements identification in a decentralized OSS development environment*. In particular, we plan to conduct this investigation in light of the theory of new idea generation widely used in sociology and anthropology [15], [31], [32]. According to such theory, diverse social interaction can help a stakeholder contribute new ideas. Therefore, we intend to consider the stakeholders and their social interaction as independent variable of this study, whereas the dependent variable is the number of new requirements proposed by the stakeholders [33]. We operationalize 'new requirements' as the features that are freshly appeared (proposed) and promptly realized (closed). These requirements are interesting because they are both new, practical, and adaptive to the task constraints [34], [35]. We start examining Firefox, Mylyn [36], and CONNECT Gateway [13], three large scale OSS systems in our study. In order to support our operationalization of new requirements, we intend to investigate requirements that have been proposed and realized between releases.

Expected Outcomes: The outcomes of this study are expected to provide insights on stakeholders' social interaction and new idea generation. The preliminary analysis gives us promising results. For example, we notice that stakeholders interacting with a diverse group of people generally contribute a higher number of new requirements. In particular, 37.23%, 30.60%, and 44.83% of the new requirements for Firefox¹, Mylyn [36], and CONNECT Gateway² have been proposed by stakeholders with diverse social interactions. The detailed findings of this study are reported in [37].

B. Requirements Implementation

Requirements implementation represents the core phase in realizing requirements in a software system. It is during the implementation phase when the developers write source code that transforms the conceptual abstractions captured by requirements into deliverable features in software [35]. Requirements realization is not an isolated activity performed

by sole developers as a developer typically interacts with other stakeholders in order to clarify the technical and organizational constraints [14]. Furthermore, not all newly proposed requirements are immediately implemented by the developers. The requirements are prioritized during implementation and some complicated ones with high coding demand may be postponed for later versions [13]. As requirements implementation in a timely manner is very important in the modern deadline driven software development, developer productivity is crucial in a software system's sustainability. Research has shown that a person's interaction with others can impact her problem solving abilities [15], [38]. Despite the importance of social interaction during requirements implementation [13], [14], little is known about how social interaction among developers and other stakeholders affects developer productivity in terms of implementing the requirements.

Study Plan: Following the proposed research framework (cf. Figure 3), we plan to investigate *how stakeholders' social interaction influences developer productivity in terms of implementing requirements in the OSS development paradigm*. To that end, we plan to consider developers and their social interaction with other stakeholders as independent variable, and developer productivity as dependent variable [33]. Following existing research [39], we operationalize developer productivity as average delta per month, i.e., average number of LOC added and/or deleted by a developer per month. Our intention is to study two large and successful OSS projects, namely Firefox¹ and Mylyn [36]. Existing research suggests that social interaction can lead to improved use of available information, potentially providing a positive effect on a person's productivity [15], [38]. Thus, we anticipate this study will provide interesting findings that can help software practitioners better manage available resources to address time critical requirements.

Expected Outcomes: The outcomes are expected to shed some light on the impact of stakeholders' social interaction on their productivity. The study is currently in progress and we plan to report the results in details in one of our upcoming publications.

C. Creativity in RE

Creativity, defined by Sternberg, can be considered as "the ability to produce work that is both novel (i.e. original and unexpected) and appropriate (i.e. useful, adaptive to task constraints)" [34]. According to Maiden *et al.* [35], creativity in RE is the capture of requirements that are new to the project stakeholders but may not be historically new to humankind. Creative requirements could be obtained by exploring, combining, and transforming existing ideas in the conceptual space [40] that includes problem and machine domains in RE [4], [5]. In this way, creativity in RE can be considered as a product of the intertwined nature of the "Twin Peaks" that allows us to conceptualize swinging and sharing ideas and experiences between the domains (cf. Figure 3). Thus, creativity is placed in between the "Twin Peaks" signifying the importance of both application and implementation domains.

Creativity in RE has been identified to be a collaboratively creative process where collaboration enables exposure to diverse ideas [41]. In an RE environment, therefore, stakeholders' social interaction works as a means to interchange knowledge and concepts between the "Twin Peaks" that can be explored, combined, or transformed to obtain creativity requirements. Techniques, such as generating requirements with scenario, have been proposed to support creativity while exploring information analogical to the current context [42], [43]. To further advance the knowledge of creativity in RE, we plan to investigate stakeholders' social interaction from a combinational creativity perspective, i.e., making unfamiliar connections between familiar possibilities of requirements [44].

Study Plan: We propose to examine *how stakeholders' social interaction can be utilized to achieve combinational creativity in RE*. To that end, we plan to mine existing requirements and associated stakeholder discussions and obtain familiar ideas in terms of dominant topics [45]. In order to find unfamiliar connections between these familiar ideas to generate new requirements, we plan to exploit linguistic techniques, such as part-of-speech (POS) tagging [46]. For example, we can use POS tagging to identify the most common objects and the most common verbs from the topics commonly discussed by stakeholder groups. Flipping the POSs can provide us unfamiliar verb-object combinations that can then be elaborated to obtain new requirements. To further assess the creativity merit of these requirements, we propose to conduct a human subject study considering the requirements as independent variable and the creativity ratings provided by the humans as dependent variable. We consider Firefox¹ and Mylyn [36] as the subject systems and mine existing requirements and stakeholders' comments in order to create new requirements.

Expected Outcomes: Among the expected outcomes of this study are the new requirements for subject systems with promising average creativity ratings provided by human analysts. For instance, the words 'text' and 'number' are found to be a common object and a common verb respectively and flipping the POSs gives us an unfamiliar verb-object combination 'text-number' in the context of Firefox¹. Elaborating this combination, we obtain a Firefox requirement in our study "Firefox user can text phone number from the web page" with an average creativity rating of 3.5 on a 5-point Likert scale [47]. The findings of this study are detailed in [48].

IV. PROGRESS

So far, some progresses have been made following the research framework proposed in this paper. In particular, we have conducted our planned studies on requirements identification and creativity in RE. The detailed findings from these studies are reported in papers [37] and [48], submitted to a journal and the RE 2014 conference respectively, that are currently under review.

Our study investigating the role of stakeholders' social interaction in identifying new requirements has revealed that generally social interaction positively impact new requirements

identification [37]. Further interesting finding from this study, however, is that stakeholder's role together with social interaction plays an important part in this context [37]. These findings attest stakeholders' social interaction as a weaving mechanism of the "Twin Peaks" in open-source RE. Furthermore, this work advances the fundamental understanding about the interplay between stakeholders' organization in their social network and the attributes of the software artifacts contributed by them.

Our study on exploiting stakeholders' social interaction to facilitate combinational creativity in RE [44] has also shown promising results [48]. Following the research plan outlined in Section III-C and detailed in [48], we were able to obtain several new requirements for both Firefox¹ and Mylyn [36] where the human subject study revealed encouraging results about the novelty of these requirements. The outcomes of this study not only stress stakeholders' social interaction as a weaving mechanism but also advance the current solutions that facilitate creativity practice in the RE process of OSS systems.

The proposed study on stakeholders' social interaction and productivity is currently in the data collection and analysis phase. Based on the initial trial analysis conducted on the collected data, we are optimistic about obtaining valuable insights about both the "Twin Peaks" model and the open-source RE process from this study as well. We plan to wrap up the current phase of the study in a couple of months and intend to report our detailed findings in a journal article.

V. SUMMARY AND FUTURE WORK

In this research proposal, we have formulated a central hypothesis to guide our overall research. The contributions of our work lie in the novelty of the hypothesis that exploits the interaction among the stakeholders as an underlying mechanism to weave a variety of human-centered activities in the "Twin Peaks" of modern open-source RE. Guided by the unified conceptualization, we have developed concrete research plans to test the central hypothesis via three activities: requirements identification, requirements implementation, and creativity in RE. Carrying out the research plans will enable us to gain valuable insights, which we will use to update the assumptions if necessary and to generate guidelines for enhancing software engineering practice in relevant areas.

Our future work includes further examining stakeholders' social interaction from other RE activities, such as requirements prioritization and negotiation. We also plan to conduct further studies to test our hypothesis in commercial software development environment thereby pushing the knowledge from our overall research towards the software development paradigm in general.

ACKNOWLEDGMENT

I express my sincere and deepest gratitude to my advisor, Dr. Nan Niu for his kind support and guidance throughout this research. The research is partially supported by the U.S. NSF (National Science Foundation) Grant CCF-1238336.

REFERENCES

- [1] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00, 2000, pp. 35–46.
- [2] M. Jarke, P. Loucopoulos, K. Lyytinen, J. Mylopoulos, and W. Robinson, "The brave new world of design requirements," *Information Systems*, vol. 36, no. 7, pp. 992–1008, 2011.
- [3] B. Nuseibeh, "Weaving together requirements and architectures," *IEEE Computer*, vol. 34, no. 3, pp. 115–119, 2001.
- [4] M. Jackson, "The meaning of requirements," *Annals of Software Engineering*, vol. 3, no. 1, pp. 5–21, 1997.
- [5] ———, "Problems and requirements," in *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering (RE)*, 1995, pp. 2–8.
- [6] M. Jackson and P. Zave, "Deriving specifications from requirements: an example," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 1995, pp. 15–24.
- [7] D. Mou and D. Ratiu, "Binding requirements and component architecture by using model-based test-driven development," in *Proceedings of the 1st International Workshop on the Twin Peaks of Requirements and Architecture (Twin Peaks)*, 2012, pp. 27–30.
- [8] M. S. Loft, S. S. Nielsen, K. Nørskov, and J. Jorgensen, "Interplay between requirements, software architecture, and hardware constraints in the development of a home control user interface," in *Proceedings of the 1st International Workshop on the Twin Peaks of Requirements and Architecture (Twin Peaks)*, 2012, pp. 1–6.
- [9] J. Cleland-Huang, "Meet elaine: A persona-driven approach to exploring architecturally significant requirements," *IEEE Software*, vol. 30, no. 4, pp. 18–21, 2013.
- [10] N. Niu, L. Xu, J.-R. Cheng, and Z. Niu, "Analysis of architecturally significant requirements for enterprise systems," *IEEE Systems Journal (Accepted)*, 2013.
- [11] T. A. Alspaugh and W. Scacchi, "Ongoing software development without classical requirements," in *Proceedings of the International Conference on Requirements Engineering (RE)*, 2013, pp. 165–174.
- [12] X. Xiao, A. Lindberg, S. Hansen, K. Lyytinen *et al.*, "computing-requirements in open source software projects," in *Proceedings of the International Conference on Information Systems (ICIS)*, 2013.
- [13] N. A. Ernst and G. Murphy, "Case studies in just-in-time requirements analysis," in *Proceedings of the International Workshop on Empirical Requirements Engineering at RE*, 2012, pp. 25–32.
- [14] L. Panjer, D. Damian, and M. Storey, "Cooperation and coordination concerns in a distributed software development project," in *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. ACM, 2008, pp. 77–80.
- [15] P. Pirolli, "An elementary social information foraging model," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2009, pp. 605–614.
- [16] M. Mirakhorli and J. Cleland-Huang, "Traversing the twin peaks," *IEEE Software*, vol. 30, no. 2, pp. 30–36, 2013.
- [17] I. Ozkaya, L. Bass, R. L. Nord, and R. S. Sangwan, "Making practical use of quality attribute information," *IEEE Software*, vol. 25, pp. 25–33, 2008.
- [18] W. Scacchi, "Understanding the requirements for developing open source software systems," in *IEEE Software*, vol. 149, no. 1, 2002, pp. 24–39.
- [19] W. Scacchi, C. Jensen, J. Noll, and M. Elliott, "Multi-modal modeling of open source software requirements processes," in *Proceedings of the International Conference on Open Source Software*, 2005, pp. 1–8.
- [20] P. Pirolli, *Information foraging theory: Adaptive interaction with information*. Oxford University Press, 2007.
- [21] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT/FSE)*, 2008, pp. 24–35.
- [22] M. Cataldo and J. Herbsleb, "Communication networks in geographically distributed software development," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. ACM, 2008, pp. 579–588.
- [23] T. Wolf, A. Schroter, D. Damian, and T. Nguyen, "Predicting build failures using social network analysis on developer communication," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2009, pp. 1–11.
- [24] R. K. Yin, *Case study research: Design and methods*. SAGE Publications, 2008, vol. 5.
- [25] I. Sommerville and G. Kotonya, *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc., 1998.
- [26] D. Zowghi and C. Coulin, "Requirements elicitation: A survey of techniques, approaches, and tools," *Engineering and Managing Software Requirements*, pp. 19–46, 2005.
- [27] E. Kavakli, "Goal-oriented requirements engineering: A unifying framework," *Requirements Engineering Journal (REJ)*, vol. 6, no. 4, pp. 237–251, 2002.
- [28] J. P. Rodríguez, C. Ebert, and A. Vizcaino, "Technologies and tools for distributed teams," *IEEE Software*, vol. 27, no. 5, pp. 10–14, 2010.
- [29] R. Agarwal and M. R. Tanniru, "Knowledge acquisition using structured interviewing: an empirical investigation," *Journal of Management Information Systems*, pp. 123–140, 1990.
- [30] N. Niu and S. M. Easterbrook, "So, you think you know others' goals? a repertory grid study," *IEEE Software*, vol. 24, no. 2, pp. 53–61, 2007.
- [31] R. S. Burt, *Brokerage and closure: An introduction to social capital*. Oxford University Press, USA, 2005.
- [32] J. N. Cummings, "Work groups, structural diversity, and knowledge sharing in a global organization," *Management science*, vol. 50, no. 3, pp. 352–364, 2004.
- [33] D. Montgomery and G. Runger, *Applied statistics and probability for engineers*. Wiley, 2010.
- [34] R. J. Sternberg, *Handbook of creativity*. Cambridge University Press, 1999.
- [35] N. Maiden, S. Jones, K. Karlsson, R. Neill, K. Zachos, and A. Milne, "Requirements engineering as creative problem solving: A research agenda for idea finding," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2010, pp. 57–66.
- [36] M. Kersten and G. Murphy, "Using task context to improve programmer productivity," in *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT/FSE)*, 2006, pp. 1–11.
- [37] T. Bhowmik, N. Niu, and P. Singhania, "On the role of structural holes in requirements identification: An exploratory study on open source software development," *ACM Transactions on Management Information Systems (TMIS) Special Issue on Complexity Systems Evolution: Requirements Engineering Perspective*, (Major revision under review).
- [38] S. Clearwater, T. Hogg, and B. Huberman, "Cooperative problem solving," *Computation: The Micro and the Macro View*, pp. 33–70, 1992.
- [39] A. Mockus and J. Herbsleb, "Expertise browser: a quantitative approach to identifying expertise," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2002, pp. 503–512.
- [40] M. A. Boden, *The creative mind: Myths and mechanisms*. Routledge, 2003.
- [41] M. Mahaux, L. Nguyen, O. Gotel, L. Mich, A. Mavin, and K. Schmid, "Collaborative creativity in requirements engineering: Analysis and practical advice," in *Proceedings of the International Conference on Research Challenges in Information Science (RCIS)*, 2013, pp. 1–10.
- [42] I. K. Karlsson, N. Maiden, and A. Kerne, "Inventing requirements with creativity support tools," in *Requirements Engineering: Foundation for Software Quality*. Springer, 2009, pp. 162–174.
- [43] K. Zachos and N. Maiden, "Inventing requirements from software: An empirical investigation with web services," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2008, pp. 145–154.
- [44] N. Maiden, "Requirements engineering as information search and idea discovery (keynote)," in *Proceedings of the International Requirements Engineering Conference (RE)*, 2013, pp. 1–1.
- [45] E. Linstead, C. Lopes, and P. Baldi, "An application of latent dirichlet allocation to analyzing software evolution," in *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, 2008, pp. 813–818.
- [46] E. Brill, "A simple rule-based part of speech tagger," in *Proceedings of the Workshop on Speech and Natural Language*, 1992, pp. 112–116.
- [47] R. Likert, "A technique for the measurement of attitudes," *Archives of psychology*, 1932.
- [48] T. Bhowmik, N. Niu, A. Mahmoud, and J. Savolainen, "Ongoing software development without classical requirements," in *The International Conference on Requirements Engineering (RE)*, 2014 (Under review).

Aligning Services and Requirements with User Feedback

Muneera Bano

Research Centre for Human Centered Technology Design

Faculty of Engineering and IT

University of Technology, Sydney

Muneera.Bano@student.uts.edu.au

Abstract—For analysts the alignment between the requirements and the available services presents a significant challenge in service oriented paradigm. To address this challenge various technical solutions have already been proposed. Although technical issues play an important role in this selection but organizational and social factors are equally as important in selecting an optimally aligned service for a specific requirement. The users of services are mostly ignored in the alignment process. User feedback analysis has recently gained a lot of research focus, but these benefits have not been fully explored and utilized in service oriented software development. In this paper I present a method for aligning services to requirements that is designed using the Situational Method Engineering approach and it incorporates user feedback about the services. This feedback assists the analysts in extracting required information for making informed decisions while selecting services among available options that satisfies both the user requirements and customer preferences. The method is supported by a proposed tool. The method and the supporting tool will be validated by a controlled experiment and focus group feedback from the practitioners.

Index Terms—Users, Requirements, Services, Alignment

I. INTRODUCTION

Service Oriented Software Engineering (SOSE) aims to reuse existing software components in the form of services to reduce the time, cost and effort of the development [1]. In spite of being advocated as a new architectural style of software development, SOSE has inherited the challenges of component based software development and traditional development especially in requirements engineering (RE) phase [1, 2]. In the Service Oriented Requirements Engineering (SORE) there is an additional task of alignment of services to requirements. By aligning a service to the requirements I refer to finding the best matched service for the requirements while making a tradeoff among cost, functional and non functional requirements.

Many solutions have been proposed to tackle alignment from technical perspective in service oriented domain (e.g. model based transformation [3], iterative process [4], granularity analysis [5-7]) alike. But according to our industrial interviews, practitioners are facing more challenges when dealing with users and customers rather than when dealing with technical solutions [2, 8]. Technical perspective of any issue is just a single view of a multifaceted problem [9, 10]. The technology is important, but how to make the best use of that technology to satisfy those who will ultimately use the system,

is far more crucial. Without users' satisfaction and approval the results would be considered a failure no matter how advance the technology would become. There is a need for further exploration on the issue of alignment in SORE [11]. According to my qualitative industrial interview study with practitioners [2, 8], alignment of requirements to the services was pointed out as most challenging task due to the following reasons:

1. The service specifications are sometimes not at the same level of abstraction to the requirements in describing the details of functionality offered by the service. In some cases, there might be some crucial information missing in the service specification specially related to the quality attributes that might be important in customer requirements for service selection.
2. The service granularity (range of functionality offered by a service [12]) may not be at the same level as requirements. Services can either be fine-grained (focused, limited functionality) or coarse-grained (broader functionality). Fine granularity offers more flexibility and reusability in customizing the system but also results in increasing the effort and cost of integration [13]. A coarse-grained service is typically expected to carry out more functions but would also exchange more messages and data, and may present a complex interface [5] and reduces the reusability as the functionality might be too overloaded for the actual need of most consumers [6]. It is challenging for the analysts to find a service at the right level of granularity as it requires a trade-off analysis between cost and functional aspect as well as various facets of flexibility, reusability and performance of the service to name a few.
3. The services are typically developed generic and context free to make them reusable and also to suite the needs of a wide range of customers. In some cases practitioners find it difficult to align context free services to the projects where the contextual details were crucial.
4. The practitioners' most frequently mentioned reason for challenges in alignment is the dissatisfaction of the customers and the users with the system being developed. Currently in service selection process, there is a lack of a proper mechanism for reusing the experience of the people who have used a service in the past. The users of a service have the real life experience of actually using the service and hence their feed back can provide very useful information for the service selection process. User

feedback involvement in system development and decision making processes has been claimed to increase users' satisfaction with the end product [14], but the benefits of the concepts of user involvement have not been fully explored and utilized in SOSE.

The current solutions for the alignment (e.g. [3-5, 12, 13, 15]), are focused primarily on the technological aspect of the problem. Involving the people who have the past experience of using a service can provide valuable information that can be helpful in decision making process. The online social media presents a new opportunity for monitoring trends and sentiments of users towards a particular service among a large pool of options. In this paper I propose to explore the benefits of user feedback by designing a method that can help analysts in making better informed decisions to overcome the challenges of alignment during service selection [16]. The construction of the proposed method is grounded in the findings and systematic analysis of the literature [1, 14, 17, 18], a quantitative study (online survey) [11], and qualitative study (interviews with practitioners) [2, 8].

II. PROPOSED SOLUTION

“Users’ involvement” has been the focus of research over three decades and has been studied in various disciplines from different perspectives and has been intuitively and axiomatically accepted to play a positive key role in users’ satisfaction and ultimately leading to the system success [14, 17, 18]. In service oriented paradigm, active user involvement is needed in order to provide personalized systems that can be customized for individual user needs [19]. There are various levels, degrees and roles proposed in literature for user involvement e.g. consultative, informative, participative etc. [17, 20]. In consultative and informative role, the users are required to provide the necessary information that can impact the design related decision processes of the system development.

The user comments and feedback has been a major source of evolution of Android market and Apple store applications [21-23]. Recently online user feedback and sentiment analysis has gained a lot of interest in various areas of software engineering research e.g. Requirements Elicitation [19, 21, 24], Software Evolution [25, 26], Software Quality [23]. Sentiment analysis (opinion mining) is used for calculating and monitoring the attitude and behavior of the users’ feedback, comments and reviews available on the online social media. Various Sentiment Analysis tools, techniques and methods [27], are proposed that make use of NLP, Computational Linguistics, Text mining and analytics capabilities for calculating quantitative values of various users’ attitude and behavior towards a particular phenomenon or a product. The benefits of the user feedback and sentiment analysis are not being fully utilized in the alignment process of service oriented development. In my proposed method, I plan to explore the benefits of user feedback analysis on the process of alignment and to evaluate whether it helps analysts in overcoming the challenges of alignment. Figure 1 shows the high level view of

my proposed method [16]. The method has to satisfy the following two major goals:

1. It should provide a mechanism to involve the user feedback during analysis for alignment of services to requirements.
2. The method should be flexible and adaptable to suit various project contexts and situations.

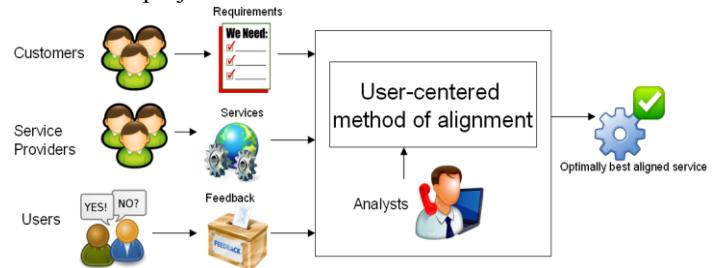


Fig. 1. Proposed Method for Alignment of Requirements and Services [16]

The best aligned service here is defined as the one that satisfies maximum set of customer requirements (both functional and non-functional) within the approved budget. Evaluating a service only with the information advertised by the service providers in their specification may not always lead to an optimal decision for service selection. User feedback can help in increasing the knowledge of analysts about actual working of the service besides the functionality published in the service specification. The proposed method will be involving both customers and users as defined below:

Customer: *for whom the service based software system is being developed who have supplied the requirements, who are the project sponsors, and will actually use this system in future.* As with any software development projects, the customers in the project participate in various activities like for requirements elicitation, modification, and prioritisation based on their preferences. For making decisions based on customer preferences in my method, I will be following the Multi Criteria Decision Analysis (MCDA) [28] approach. MCDA is used for decision making in situations where a trade-off is required among multiple criteria. This approach has been extensively used in Management Sciences for the selection of suppliers. It helps the analysts in making more informed decisions for service selection based on the perceived relative importance of customer requirements.

User: *who has experience of using a particular service in the past and has either provided feedback on online resources or can provide (post deployment) feedback when requested.* This group can include the analysts, developers, designers who have past experiences of using a particular service. Their feedback is either collected from online resources (if available) or elicited directly from the users (if approachable). In recent years, there has been a substantial body of research for proposing methods, tools and techniques on collecting and analyzing users’ feedback for extracting useful information from the extensive user comments and feedback available on online resources [19, 21, 24-26, 29, 30] (e.g. sentiment analysis, opinion mining, information retrieval, crowd sourcing, parsing, natural language processing). The user feedback can help the analysts in alignment process with (1)

finding out the missing information in service specification, (2) collecting contextual data based on the previous use of the service, (3) eliciting past users' satisfaction level with the service and the reputation of service providers, and (4) comparing the service specification details against the real use of the service. The benefits of user feedback analysis can help in increasing the information for analysts in making more informed decision for service selection.

III. RESEARCH METHODOLOGY

My research is exploratory in nature and is based on the following research question:

Does involving user feedback in alignment of requirements and services leads to the better service selection?

The 'better service selection' is measured based on the coverage of the functional and non-functional requirements by the proposed method in comparison with the alignment methods that does not include user feedback. The research methodology that I have followed for my thesis has three main stages: Defining Problem Space, Solution Space and Empirical Validation (Fig 2).

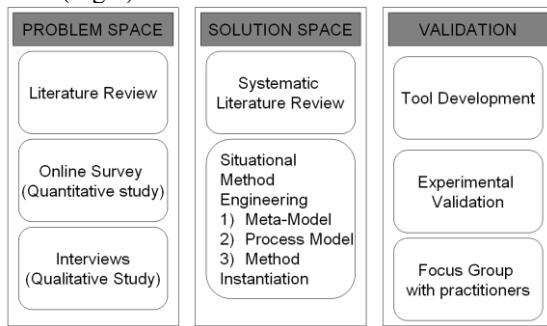


Fig. 2. Research Methodology

A. Problem Space

In order to define the problem and scope of the research, I have conducted a comprehensive literature review to find the challenges of SORE reported in the published literature [1]. Based on the results of literature review I conducted an online survey for confirmation of those challenges from 117 practitioners across the globe having experience of working on service oriented projects [11]. Utilizing the results from the literature review and survey I conducted open ended interviews with 14 experts from industry in Australia to gain a detailed insight into the challenges of service oriented requirements engineering [2, 8]. The alignment between business requirements and services was identified as the top challenge that business analysts have to face in service oriented projects.

B. Solution Space

1) *Systematic Literature Review:* As I was interested in exploring the concepts of user involvement in alignment process, I conducted a systematic literature review of the existing empirical literature on the topic. The review resulted in 87 studies from the literature for the period of 1980-2012 [14, 17, 18]. The results show a positive impact of user

involvement in terms of system success from empirical literature.

2) *Situational Method Engineering (SME):* Due to the dynamic nature of software development, one fix methodology for any process would not work in every context and domain. SME provides a systematic and flexible approach to tailor methods for software development activities according to project requirements at run time [31]. It requires availability of the method base (MB) for the project team which is a kind of a lookup table where different methods (method chunks) are listed along with their associated situational factors (when to use them), and required tools/techniques to carry them out. The MB is dynamic and can grow with the new projects and methods being introduced. According to the project context and situations the suitable method chunks are selected by following the process guidelines and combined to tailor a method for the project at hand [32]. Software development in service oriented domain is dynamic in nature and SME is an appropriate choice for building a method of alignment for a specific project context based on the available services. SME has been applied to modify the existing method of alignment by to involve customer and users. SME recommends a three tiered structure for designing methods [31]. On top level is the method meta-model that represents the components and rules for constructing method, second level has a specific instantiation of the method meta-model and on third level is an instance of the method for specific project situation.

a) *Meta-Model:* The elements in the meta-model are inspired by the the three meta-classes (Producers, Work units, work products) in ISO/IEC 24744 standard [33], which guides the development of meta-models for software engineering methods. Table I gives the definition of the elements in my proposed method's meta-model and show their mapping to the three meta-classes in ISO/IEC 24744. Figure 3 represents the meta-model of the current approaches available for alignment [3-5, 12, 13, 15], and figure 4 represent the meta-model of our proposed method by highlighting the difference with existing approaches with two new elements i.e. User Feedback and Multi Criteria Decision Analysis.

b) *Process Model:* From the meta-model, the process model for alignment method is derived (Figure 5). Besides the three actors that are typically involved in the alignment process (analyst, service provider, customer), our proposed method has an additional actor i.e. user.

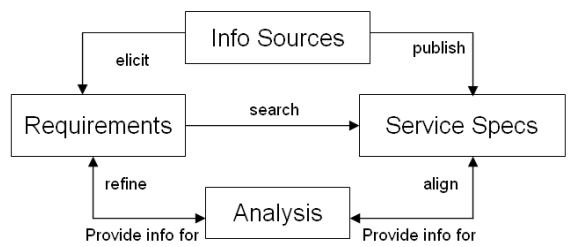


Fig. 3. The Meta Model of current methods of aligning requirements to service specification

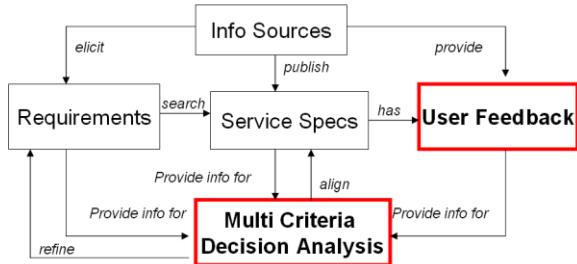


Fig. 4. The Meta Model of proposed method of aligning requirements to service specification

TABLE I. MAPPING OUR META-MODEL CLASSES TO ISO/IEC 24744

| Classes in our Meta-Model | Description of classes in our Meta-Model | ISO/IEC 24744 [33] relevant meta-classes |
|---|---|--|
| INFO SOURCES | The people or objects that provide required information or knowledge. E.g. customers, users, repositories, analysts, knowledge base | PRODUCERS: <i>Tool</i> (an instrument that helps another producer to execute its responsibilities), <i>Person</i> (allows taking into account individual persons at the endeavour level.) |
| REQUIREMENTS | Requirements for the project that are used for searching available services | WORK PRODUCTS: <i>Document</i> (a durable depiction of a fragment of reality), |
| SERVICE SPECS | Describing the functionalities offered by the services | |
| USER FEEDBACK | Provided by actual users of the service, (1) either available over the web or (2) elicited | |
| MULTI CRITERIA DECISION ANALYSIS | Analysing the services based on customer preferences | WORK UNITS: <i>Process</i> (large-grained, operating within a given area of expertise), <i>Technique</i> (small-grained, focusing on how the given purpose may be achieved). |

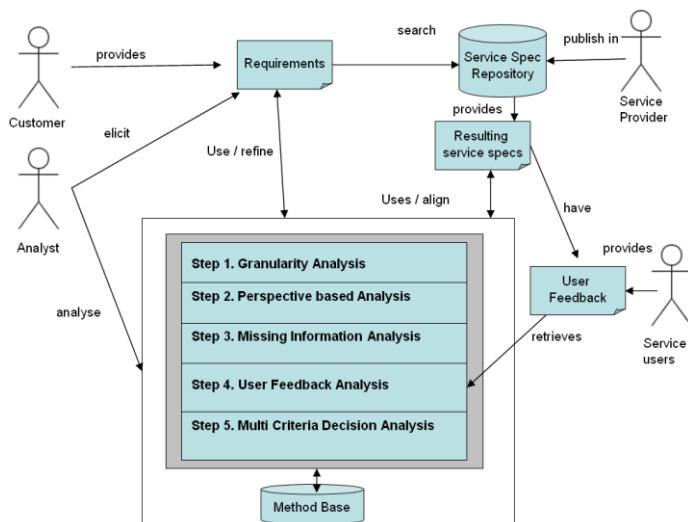


Fig. 5. Proposed Method for Aligning services to requirements [16]

The process alignment starts elicitation of an initial set of requirements from customers represented by \mathbf{R} such that $\mathbf{R} = \{R_1, R_2, R_3 \dots R_X\}$ where X is the total number of requirements. Using the requirement set the analyst would search for available related services from accessible service repositories (local or global). Resulting services can be represented by $\mathbf{S} = \{S_1, S_2, S_3 \dots S_Y\}$ where Y is number of services found against requirement set \mathbf{R} . The analysis in the proposed method comprises of five interconnected and iterative steps: Granularity Analysis, Perspective based Analysis, Missing information Analysis, User Feedback Analysis, and Multi Criteria Decision Analysis.

Granularity Analysis: The first step requires the analysts to evaluate all service specifications for granularity level against requirements and score them for their level of granularity. The aim is to select the service that provides maximum functional range against requirements, i.e. a service that provides more coverage of requirement set. The scores can be calculated by evaluating a service in one of the three following scenarios: fully aligned (score 1), totally misaligned (score 0), or partially aligned (score between 0 to 1). Partial alignment has further three cases for the analyst to evaluate before selecting the service; (1) Service functionality fulfils only part of the requirement (service is too fine grained causing performance related issues e.g. Integration problem, delays in interaction among multiple services), (2) Service functionality offers more than the requirements specifies (service is too coarse grained increasing cost), (3) Service functionality and the requirements overlap (causing increase in cost and integration issues).

Considering the format for service specification various methods are available (e.g. [5-7] for determining the level of granularity of service against requirements. The granularity method is represented by **G-Method** which is selected from MB according to the situation (e.g. format of service specification) to calculate the functional range of service specifications against requirement set \mathbf{R} .

$$g_i = \mathbf{G-Method}(S_i, \mathbf{R})$$

Perspective-based Analysis: During this step the requirements are to be analyzed from multiple perspectives (e.g. Organizational, Technical (Functional, non-Functional), Economical, and Project related etc.) in order to develop a checklist. The analyst converts the requirements into perspective based checklist and assigns the weights to the checks based on customer preferences. The set of checks is represented by $\mathbf{C} = \{C_1, C_2, C_3 \dots C_K\}$ and the weights against these checks is represented by $\mathbf{W} = \{W_1, W_2, W_3 \dots W_K\}$ where K is the number of checks in the perspective based checklist. The analyst evaluates service specification to provide answers to the checks in the checklist (from 0 to 1). For a service S_i from the set of services \mathbf{S} the perspective based score is represented by P_i which is calculated by adding all the answers to the K number of checks in set \mathbf{C} for that service according to the following formula.

$$P_i = \sum_{i=1}^K ((c_i) * (w_i))$$

Missing Information Analysis: While aligning the services against requirements, there is a possibility that some

information (especially performance related) might be missing in service specification. This can be due to the mismatch in the level of abstraction in describing the service functionality in specification. The analyst is required to assess the missing information (for its type and context) because it will be extracted from user feedback in the next step. Suitable methods are selected from MB for this purpose (e.g. feature extraction, information retrieval, survey questionnaire etc). If the information can not be retrieved from any sources (e.g. blog, forums, surveys), in MCDA there are methods for imprecise and incomplete information that can be applied e.g. sensitivity analysis, fuzzy goal programming [28].

User Feedback Analysis: User feedback can be collected based on two scenarios: (1) directly from online resources if available or (2) elicited directly from the users if approachable. There are various methods and associated tools/techniques available for feedback collection based on the situations and format in which the feedback is available (e.g. sentiment analysis, feature extraction, information retrieval, crowd sourcing, survey and questionnaire etc.). This feedback is serving two main purposes in my method: (1) providing the information about the users' satisfaction based on their past experience of using the service. This will also reflect users' trust of service provider (if the service is from third party), (2) Service specification may not be at the same level of abstraction as customer requirements in giving details about functional and non functional capabilities of service user feedback is also used for filling the gaps in service specification where the information is missing [25]. Sentiment analysis tools and methods (**US-Method**) help in providing quantifiable scores for calculating user satisfaction u_i against a service by analysing the online or offline user comments, reviews and feedback.

$$u_i = \text{US-Method} (S_i)$$

Multi Criteria Decision Analysis: Once all the information is acquired (as much as possible), the appropriate MCDA method can be applied for calculating scores for all Y candidate services. Additive weighting method is the most widely used approach to deal with multi criteria decision problems [28]. It's a simple method for multiplying the weights with the scores of the checks and then adding up to see which of the service has scored the highest. This method makes the assumption that quantifiable scores for the weights in the same unit of measurement are being provided by the customer based on their prioritization of the requirements. For example in Additive Weighting method when the customer provides the preferences for the weights for Perspective based checklist w_p and the level of granularity w_g and user satisfaction w_u to be considered for final decision, the final score for a service S_i in the set S can be calculated as following.

$$\text{Score} (S_i) = (P_i * w_p) + (g_i * w_g) + (u_i * w_u)$$

When the scores are calculated for all Y number of services, the highest service score among the set S will be the optimally the best aligned service according to the customer preferences.

In cases where this may not be possible, Aspiration level Methods [28] are available where the preferences are

considered in their natural way rather than converting them all into one scoring level. Another more dynamic (on the run) approach is Outranking Method [28], which helps in construction of the preference based checklist by considering and evaluating different available alternative at the time of decision, rather than before the analysis.

3) *Instantiation:* I am currently analysing data that I have collected by instantiating my method on a real case of a service selection. The aims of this case study are twofold (1) refinement and improvement of my proposed method, and (2) preliminary validation of the idea of involving user feedback in the service selection process. The study involved the selection of an SMS gateway service for an existing website that belongs to a gym. Online searches resulted in 92 eligible options providing the SMS gateway services. From the customer requirements 28 checks were created for evaluating these services. This created a complex and challenging scenario for decision making as many of the services offered more or less the same functionality within the same price range. The requirements became further refined and more complete later when I was analyzing service specification with better understanding of the functionalities of the available services. For user feedback analysis I used a free online sentiment analysis tool socialmention.com. The preliminary results of my study have shown that in scenarios with significant number of services, it is helpful for analysts to consider additional information to select optimally aligned service to the requirements.

C. Validation

This stage of the research methodology is currently in progress. In this last phase we have three steps:

1) *Tool development:* By utilizing the design of the alignment method described in this paper, I am currently in the process of designing and developing an automated web based tool to support the analysts and customers in analysis and decision making by simulating the different scenarios of MCDA scoring methods.

2) *Experimental Validation:* In this step the method and the supporting tool will be validated in an academic experimental lab set up to evaluate the effectiveness of our proposed method. The hypothesis for this experiment is that user feedback will help the analysts in making better decision in alignment of services to requirements. The experiment will be conducted on two service oriented projects. Both projects will be developed in two instances; one using the proposed method and one without the method. The main criteria for the measurement of the effectiveness of the method are based on the coverage of the functional and non functional requirements. An additional survey questionnaire will be administered with the customers of the project to evaluate their level of satisfaction with the results from both instances of the two projects.

3) *Focus group with practitioners:* The web based tool and the results from the experiment will be presented to the practitioners (including the ones who already helped in problem definition and solution formulation) for feedback and

review. The aim is to triangulate the results of the experiments and to gain feedback from the people from industry through focus group method, which is a cost effective way of collecting timely feedback on new results.

IV. CONCLUSION AND PROGRESS

In this paper I have presented my extended proposal for a method of aligning service specification to requirements while exploring the benefits of user feedback [16]. The method is designed following Situational Method Engineering guidelines to make it flexible for adoption in various project situations. The method will be supported by a web-based tool and will be validated through a controlled experiment and focus group method. This method is based on my ongoing research driven by rigorous and systematic literature reviews [1, 14, 17, 18], a quantitative study (using online survey) [11], and a qualitative study (through interviews with practitioners) [2, 8]. Currently I am working on the design of the supporting tool that will be used in the empirical validation phase.

ACKNOWLEDGMENT

I would like to present my acknowledgement to The International Schlumberger Foundation Paris for funding my PhD and research through Faculty for the Future Award Program 2014-2015.

REFERENCES

- [1] M. Bano, and N. Ikram. Issues and challenges of requirement engineering in service oriented software development. in Fifth International Conference on Software Engineering Advances (ICSEA), 2010: IEEE.
- [2] M. Bano, D. Zowghi, N. Ikram, and M. Niazi, 'What makes Service Oriented Requirements Engineering challenging? A qualitative study', IET Software, 2014.
- [3] A. Gehlert, N. Bramsiepe, and K. Pohl. Goal-driven alignment of services and business requirements. in International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements, SOCCER 2008: IEEE.
- [4] K. Zachos, N. Maiden, and R. Howells-Morris, Discovering web services to improve requirements specifications: does it help?, in Requirements Engineering: Foundation for Software Quality. 2008, Springer. p. 168-182.
- [5] C. Steghuis, "Service granularity in SOA projects: a trade-off Analysis", Master's thesis, University of Twente, 2006.
- [6] B. Heinrich, Henneberger, M., Krammer, A., & Lautenbacher, F., Granularity of services—an economic analysis. Business & Information Systems Engineering, 2011. 3(6): p. 345-358.
- [7] N. Kulkarni, and V. Dwivedi. The role of service granularity in a successful SOA realization a case study. in IEEE Congress on Services-Part I, 2008. IEEE.
- [8] M. Bano Sahibzada, and D. Zowghi. Service Oriented Requirements Engineering: Practitioner's Perspective. in Service-Oriented Computing-ICSOC 2012 Workshops. 2013: Springer.
- [9] I. Mitroff, and H.A. Linstone, The unbounded mind: Breaking the chains of traditional business thinking. 1993: Oxford University Press.
- [10] D. Kunda, and L. Brooks. Applying social-technical approach for COTS selection. in Proceedings of the 4th UKAIS Conference. 1999.
- [11] M. Bano, and N. Ikram. KM-SORE: knowledge management for service oriented requirements engineering. in The Sixth International Conference on Software Engineering Advances ICSEA 2011.
- [12] M. P. Papazoglou, and W.-J. Van Den Heuvel, Service-oriented design and development methodology. International Journal of Web Engineering and Technology, 2006. 2(4): p. 412-442.
- [13] S. Adam, N. Riegel, and J. Doerr. Deriving software services from business processes of representative customer organizations. in International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements, SOCCER 2008: IEEE.
- [14] M. Bano, and D. Zowghi. User involvement in software development and system success: a systematic literature review. in Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, EASE. 2013: ACM.
- [15] M. Galster, and E. Bucherer. A business-goal-service-capability graph for the alignment of requirements and services. in IEEE Congress on Services-Part I, 2008: IEEE.
- [16] M. Bano, and N. Ikram, Addressing the Challenges of Alignment of Requirements and Services: A vision for User-Centered Method. Asia Pacific Requirements Engineering Symposium (APRES 2014), Auckland, New Zealand, 2014. 432: p. 83-89.
- [17] M. Bano, and D. Zowghi, A Systematic Review on the Relationship between User Involvement and System Success. Information and Software Technology, 2014 (in press).
- [18] M. Bano, and D. Zowghi. Users' Involvement in Requirements Engineering and System Success. in Empirical Requirements Engineering (EmpiRE) at RE 2013, Brazil. 2013: IEEE.
- [19] N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. in 18th IEEE International Requirements Engineering Conference (RE), 2010: IEEE.
- [20] L. Damodaran, User involvement in the systems design process-a practical guide for users. Behaviour & information technology, 1996. 15(6): p. 363-377.
- [21] L.V. Galvis Carreño, and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. in Proceedings of the 2013 International Conference on Software Engineering. IEEE Press.
- [22] B. Fu, Lin, J., Li, L., Faloutsos, C., Hong, J., & Sadeh, N. Why people hate your app: making sense of user feedback in a mobile app store. in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013: ACM.
- [23] M. Chen, and X. Liu. Predicting popularity of online distributed applications: iTunes app store case analysis. in Proceedings of the 2011 iConference. 2011: ACM.
- [24] N. Seyff, F. Graf, and N. Maiden. End-user requirements blogging with iRequire. in Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2. 2010: ACM.
- [25] D. Pagano, and B. Bruegge. User involvement in software evolution practice: a case study. in Proceedings of the 2013 International Conference on Software Engineering. 2013: IEEE Press.
- [26] D. Pagano, and W. Maalej. User feedback in the appstore: An empirical study. in 21st IEEE International Requirements Engineering Conference (RE), 2013: IEEE.
- [27] B. Pang, and L. Lee, Opinion mining and sentiment analysis. Foundations and trends in information retrieval, 2008. 2(1-2): p. 1-135.
- [28] R. Vetschera, Preference-based decision support in software engineering, in Value-Based Software Engineering. 2006, Springer. p. 67-89.
- [29] J. Hao, S. Li, and Z. Chen, Extracting service aspects from web reviews, in Web Information Systems and Mining. 2010, Springer. p. 320-327.
- [30] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: MSR for app stores. in 9th IEEE Working Conference on Mining Software Repositories (MSR) 2012: IEEE.
- [31] B. Henderson-Sellers, and J. Ralyté, Situational Method Engineering: State-of-the-Art Review. J. UCS, 2010. 16(3): p. 424-478.
- [32] S. Brinkkemper, M. Saeki, and F. Harmsen. Assembly techniques for method engineering. in Advanced Information Systems Engineering. 1998: Springer.
- [33] Commission, I.O.f.S.I.E., ISO/IEC 24744. Software Engineering-Metamodel for Development Methodologies, 2007.

Requirements Development and Management of Embedded Real-Time Systems

Jiale Zhou

School of Innovation, Design and Engineering

Mälardalen University

Västerås, Sweden

zhou.jiale@mdh.se

Abstract—It is well recognized that most of the anomalies, discovered in the development of embedded real-time systems, belong to requirement and specification phases. To ease the situation, many efforts have been investigated into the area. For requirements development, especially requirements validation and verification, model-driven architecture techniques can be considered as a cost-efficient solution. In order to utilize such advantages, the design of the proposed system is often specified in terms of analyzable models at the certain level of abstraction. Further, different levels of requirements are translated into verifiable queries and fed into the models to be either validated or verified. For requirements management, requirements traceability provides critical support for performing change impact analysis, risk analysis, regression testing, etc. In this thesis, we cover several topics about requirements validation, requirements verification, and requirements traceability. In particular, the technical contributions are three-fold: 1) we propose an approach to requirements validation by using the extended Timed Abstract State Machine (TASM) language with newly defined TASM constructs and, 2) we present a simulation-based method which is powered up by statistical techniques to conduct requirements verification, working with industrial applications and, 3) we introduce an improved VSM-based requirements traceability recovery approach using a novel context analysis. Further, we have demonstrated the applicability of our contributions in real world usage through various case studies.

I. INTRODUCTION

With the growing complexity of Embedded Real-Time Systems (ERTS), requirements development and management become more critical activities for developing such systems. Studies have revealed that most of the anomalies, discovered in the development of complex systems, belong to requirement and specification phases [1] [2] [3]. A *Requirement* is defined as “a statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines)” [4]. There is a common misconception that requirements are just specified at the outset of the System Development Life Cycle (SDLC). On the contrary, various levels of requirements should be specified for the corresponding phases of the SDLC, rather than a single phase. In this thesis, we follow a three-level requirements model, which consists of customer-level, system-level, and architecture-level (design-level in other words), as shown in Figure 1. In the V-model SDLC, requirements are firstly

specified at corresponding levels of abstraction and then are used for different integration and verification purposes. More detailed description of Figure 1 can be found in Section II.

Requirements Development (RD) includes, but is not limited to, the tasks of:

- eliciting, analyzing, validating and communicating customer-level requirements,
- transforming customer-level requirements into subsequent levels of requirements,
- allocating requirements to hardware, software and test cases,
- verifying requirements,
- validating the set of requirements.

Many efforts have discussed the importance of a set of well-defined requirements [5] [6] and the implications caused by ill-defined requirements [2]. Nevertheless, it is still a challenge to define a set of complete, correct, and verifiable requirements. Especially, non-functional requirements of ERTS, in terms of timing and resource consumption (i.e., power consumption in this thesis), are of great importance to be validated and verified. RD using Model-Driven Architecture (MDA) techniques [7] can remedy the situation, which can be considered as a cost-efficient solution. Typically, model-based RD has the advantages of accelerating project development, requiring less human efforts, and making it possible to detect and fix the potential defects and errors on a fairly early stage of the SDLC. In order to utilize such advantages, the design of the proposed system is often specified in terms of analyzable models at the certain level of abstraction. Further, different levels of requirements are translated into verifiable queries and fed into the models to be either validated or verified.

Requirements Management (RM) plays an important role in keeping track of requirements and handling a large amount of software development artifacts (i.e., the artifacts hereafter), such as design specification, source codes, and test cases. The quality of RM is of great importance for systems development, mattering to e.g., customers satisfaction, requirements coverage, efficient utilization of money, time and other resources. The heart of RM is Requirements Traceability (REQT), which is defined as “the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification,

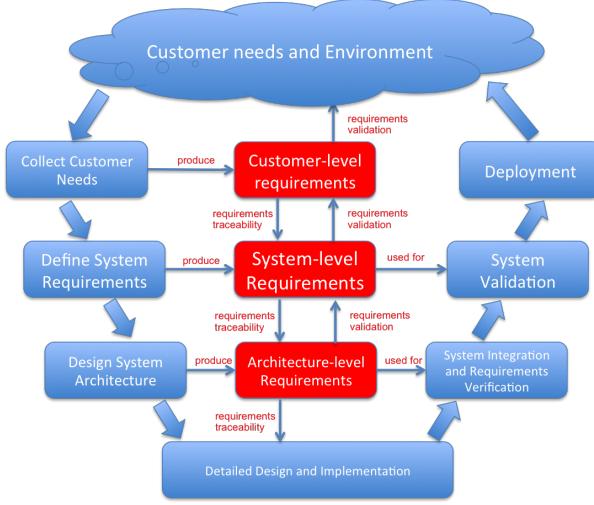


Fig. 1. The V-model SDLC and the three-level requirements model considered in this thesis.

to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases” [8]. REQT not only provides critical support for system developers throughout the entire SDLC, but also is necessarily required by the certification process. Tracing requirements can help to, but is not limited to, perform change impact analysis, risk analysis, criticality analysis, regression testing, etc. However, there are numerous challenges that make it difficult to achieve successful traceability in practice. These challenges include technical issues related to physically creating, recovering, maintaining, and using thousands of interrelated and relatively vulnerable traceability links¹. As a result, many organizations and companies are unwilling to implement and maintain traceability links, even though it is widely accepted as a critical element of the SDLC. In order to overcome the significant challenges in creating, recovering, maintaining, and using traceability, the research community has been actively addressing traceability challenges through the exploration of topics related to automating the traceability creation, facilitating the evolution and maintenance of trace links, visualizing traceability, verifying and validating systems development through traceability practices, and developing traceability practices that apply across a wide range of domains such as safety-critical applications, aspect-oriented and agile software development, as well as various regulated industries [9].

In this thesis, we cover several topics centering around RD and RM, which are mainly about requirements validation and verification, and REQT. In particular, the technical contributions are three-fold:

- We propose an approach to requirements modeling by using the Timed Abstract State Machine (TASM) language with some necessary extensions and the TASM query mechanism. To be specific, our approach describes both functional behaviors and non-functional properties,

including timing and resource consumption, of the target system, and it provides a means to requirements validation and verification by using the developed TASM Toolset and a model checker.

- We extend TASM with the capabilities of modeling non-functional properties of ERTS w.r.t timing and power consumption, using probabilistic distributions. To this end, we propose a simulation-based method which is powered up by advanced statistical techniques to conduct requirements verification. Moreover, the extended TASM inherits the easy-to-use features from the Abstract State Machine language, which is a literate specification language understandable and usable without extensive mathematical training.
- We introduce an improved VSM-based requirements traceability recovery approach using a novel context analysis. As aforementioned, in the three-level requirements model featured with customer-level, system-level and architecture-level, our method can better utilize the context information extracted from the high-level requirements to discover the subsequent artifacts at the lower levels.

II. BACKGROUND

This section firstly gives some background knowledge about requirements development centering around requirements model, requirements validation and verification in Sections II-A1 and II-A2 respectively. Next a brief overview of IR-based traceability recovery as in requirements management is presented in Section II-B.

A. Requirements Development

1) *Requirements Model*: Although there are various models for developing systems for different sizes and areas of projects [10], the development of requirements are usually divided into layers or levels which have different nuances but are similar. In this thesis, we follow the three-level requirements model, as shown in Figure 1. At the beginning of the SDLC, it is an essential task to elicit a sound set of requirements for the proposed system based on the opinions of various stakeholders, i.e., customer-level requirements. In doing so, a basis for discussion with the stakeholders and a means of clarifying their needs together with some potential solutions will be given. One example is “the customer wants a Brake-by-Wire system to substitute the traditional mechanical brake system”. Rather than proceeding straight to the design phase, it is necessary to first determine what functionalities and properties the proposed system must have, regardless of more detailed design. Therefore, the initial customer-level requirements will be further transformed, in terms of being decomposed or refined, into a set of system-level requirements, to help to establish a common understanding of the proposed solution to the domain-specific problem. One typical example is “the system shall provide a base brake functionality where the driver presses the brake pedal so that the braking system starts decelerating the vehicle”. Based off of the system-level

¹Once an endpoint of a trace link is changed, the trace link is no more valid.

requirements, it is now possible to consider alternative design architectures. The design architecture defines what components the proposed system consists of and how such components interact with each other. Subsequently, the architecture-level requirements stipulate the requirements for each system component w.r.t their functionalities, interaction constrains, and any other required properties featured with performance quality, reliability, safety, etc. One example is “the brake torque calculator shall compute the driver requested torque and send the value to the vehicle brake controller, when a brake pedal displacement is detected”. Note that, in some cases, the components at the architecture-level are still too complex to be implemented directly, and therefore will be further refined into more specific requirements corresponding to software components, hardware components which are associated with system implementation.

2) *Requirements Validation and Verification*: Requirements validation and verification are two independent activities that are performed together to ensure that the final system would meet requirements and specifications, which are imposed at the beginning of the development process. Different from system validation which confirms that the built system does what it is supposed to do, requirements validation is the process of confirming the *completeness* and *consistency* of requirements. Further, once completeness and consistency of requirements are validated, then the *correctness* of such requirements can be achieved. To be specific:

- 1) Completeness refers to situations where a specification entails all the requirements that are regarded as necessary to satisfy the proposed system, and there is no undefined object or entity.
- 2) Consistency refers to situations where a specification contains no internal contradictions.

Each of the three-level requirements has their own validation purpose as follows:

- The customer-level requirements must be guaranteed that they collect all of the stated business objectives, clarify the needs of various stakeholders, and are easy-to-use for communication between stakeholders and developers.
- The system-level requirements provide a high-level solution to implementation of the designed system, which must be validated to be sure that they address the stated business objectives, meet the needs of stakeholders, and contain no internal contradictions.
- The validation procedure for the architecture-level requirements not only ensures that they are consistent and complete with the upper-level requirements (i.e., system-level requirements), but also makes sure the design solution is feasible, unambiguous and verifiable.

Requirements verification is the process of confirming that the designed and implemented system fully addresses documented requirements. Contrary to the purpose of requirements validation focusing on “Do the right thing”, requirements verification ensures the outcome at each phase of the SDLC with the satisfaction of the pertaining requirements, in the

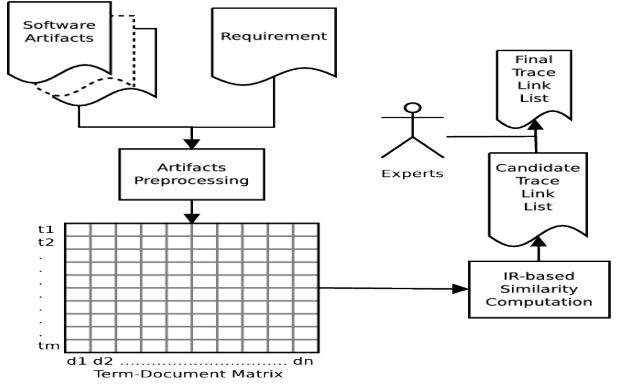


Fig. 2. An IR-based traceability recovery process.

sense of “Do the thing right”. In the model-based development, verification procedure involves modeling of the target system by using domain-specific models, which will then be analyzed by using either simulation-based methods or exhaustive analysis in terms of model checking. Among various techniques for model-based requirements verification, one interesting piece of work to mention is Statistical Model Checking (SMC) [11], which plays a more and more important role nowadays. Specifically, SMC refers to a series of techniques that observe several runs of the (system) model with respect to certain properties, and then analyze the results from the perspective of statistics, which are used to check the design correctness. There are several advantages of this application: 1) It is quite simple to implement, understand and use in practice and, 2) It requires very little or no extra modeling efforts, but simply an operational model of the target system that can be simulated w.r.t the property and, 3) The use of statistics allows to analyze non-deterministic problems. SMC gives us a strong motivation to power up our specification language, namely TASM, with advanced statistical methods.

B. Requirements Traceability

The heart of RM is building requirements traceability. The IR-based traceability recovery aims at utilizing information retrieval (IR) techniques to compare a set of source artifacts as queries (e.g., requirements), against another set of target artifacts (e.g., source code files), and to calculate the textual similarities of all possible pairs of artifacts. The textual similarity between two artifacts is based on the occurrences of terms (words) within the artifacts contained in the repository. Pairs with a similarity score lower than a certain threshold (usually defined based on engineers’ experience) are filtered out, and the reserved pairs form the candidate trace link list. The ranked list of candidate trace links are then analyzed by software engineers to decide if such links are true positive or not. Typically, an IR-based traceability recovery process follows the steps depicted in Figure 2. The artifacts have to be preprocessed before they are used to compute similarity scores. The preprocessing of the artifacts includes a text normalization by removing most non-textual tokens (e.g., operators, punctuations) and splitting compound identifiers into separate words by using the underscore or Camel Case

splitting heuristic. Furthermore, common terms, referred to as “stop words” (e.g., articles, prepositions, common usage verbs, and programming language keywords), which contribute less to the understanding about artifacts, are also discarded by using a stop word filter. Words with the length less than a defined threshold are also pruned out. In addition, stemmer is commonly used to perform a morphological analysis, which reduces the inflected words to their root, e.g., returning verb conjugations and removing plural nouns. After preprocessing, an artifact (e.g., a UC requirement, a source code file) can be represented as a plain document containing a list of terms (in this paper, we use *documents* and *artifacts* interchangeably). The extracted terms are generally stored in a $m \times N$ matrix (called term-by-document matrix), where m is the number of all the terms that occur in all the documents, and N is the number of documents in the corpus. A generic entry $w_{i,j}$ of the matrix denotes a measure of the relevance of the i_{th} term in the j_{th} document. Based on the term-by-document matrix representation, different IR methods can be used to calculate textual similarities between paired artifacts.

III. RESEARCH QUESTIONS

In this thesis, we are aiming at contributing to a general research question as follow:

- **General research question:** How to improve requirements development and management of (complex dependable) embedded real-time systems in a structured and practical way, in accordance with industrial demand?

In particular, the general research question can be further refined into three specific research questions which guide our work:

- **Research question one (RQ1):** How can we validate non-functional requirements in terms of timing and resource consumption requirements, in an early stage of model-based development of embedded real-time systems?

– **The challenge associated with RQ1 (Ch1):** In order to increase the confidence in the correctness of the requirements, model-based *formal methods* techniques have been to a large extend investigated into the field of requirements validation [12] [13]. In these techniques, the system structure and behaviors derived from low-level requirements is often specified in terms of analyzable models at a certain level of abstraction. Further, high-level requirements are formalized into verifiable queries or formulas and then fed into the models to perform model checking and/or theorem proving. In this way, the low-level requirements are reasoned about to resolve contradictions, and it is also verified that they are neither so strict to forbid desired behaviors, nor so weak to allow undesired behaviors. However, such formal methods techniques also suffer from some limitations, such as how to ease the demand of heavy mathematics background knowledge to perform theorem proving, and how to

model the target without having the state explosion problem of model checking occurred.

- **Our contribution to RQ1 (OC1):** We extend TASM with the *Observer* construct, and propose an approach to requirements modeling by using the TASM language for checking the completeness and consistency of functional and non-functional requirements at both the system-level and architecture-level.

- **Research question two (RQ2):** How can we verify system design in accordance with non-functional requirements in embedded real-time systems, featured with large-size and complicated behaviors of timing and resource consumption?

- **The challenge associated with RQ2 (Ch2):** For any embedded real-time system for instance of consumer electronics and automobiles, it should be ensured that the system design indeed meets all the specified requirements in terms of functional behaviors and non-functional properties, in a particular context. Early-stage abstract models of the system’s structure and behaviors can become the artifacts on which exhaustive analysis is usually carried out, in an attempt to increase trust in the system’s correct operation. However, the most common limitation of exhaustive analysis is the state-explosion problem which can cost unacceptable time. In particular, when the system behaves in a non-deterministic way, it is almost impossible to perform exhaustive analysis.

- **Our contribution to RQ2 (OC2):** We extend TASM with the capabilities of modeling the behaviors of systems confirming to probabilistic distributions, and then apply some statistical techniques to analyze models w.r.t functional behaviors and non-functional properties in terms of timing and resource-usage, for verification purpose.

- **Research question three (RQ3):** How can we improve the accuracy of candidate trace links between requirements and the subsequent artifacts of embedded real-time systems development?

- **The challenge associated with RQ3 (Ch3):** In traditional industrial practices, especially for legacy systems [14], trace links are manually established and maintained. Such activities tend to be costly to implement and are therefore perceived as financially non-viable by many companies [8], [15]. To address this problem, many efforts [16], [17], [18], [19], [20], [21], [22] have been devoted to semi-automatic or fully-automatic trace link creation. However, the precision and recall of generated trace links is still at a low level of accuracy, such that the trace link creation throughout the entire systems development process, remains a challenging issue [23].

- **Our contribution to RQ3 (OC3):** We improve the VSM-based requirements traceability recovery approach by using a novel context analysis. Specifically,

the analysis method can better utilize context information extracted from requirements (e.g., use cases) to discover relevant subsequent artifacts (e.g., source code files).

IV. RESEARCH METHODOLOGY

The overarching goal of our research is to improve the development process of software-intensive embedded real-time systems. Our approach focuses on extending the theoretical knowledge in the areas of requirements specification (i.e., requirements modeling in this thesis), requirements verification and validation, and requirements traceability. In order to adequately address the questions listed above, it is important to adopt an appropriate research methodology, suitable for such a given setting. The methodology used in our research is based on the research steps proposed by Shaw [24], which is summarized in the following four steps:

- 1) Conducting literature reviews and surveys to form new research goals in specific.
- 2) Analysis of the current state-of-the-art progress in requirements development and management field based on the defined research questions.
- 3) Answering the research questions by presenting the proposed solutions and achieved research results.
- 4) Validating whether the research results can be applied in the real-world applications.

V. PAPERS

In this section we provide the included papers in this thesis. Table I presents the relations between the research questions and papers.

TABLE I
RELATIONS BETWEEN THE PAPERS AND RESEARCH QUESTIONS.

| Paper | Research Question | Current State |
|---------|-------------------|----------------------------|
| Paper A | RQ1 | Published at Ada-Europe'14 |
| Paper B | RQ1 | Accepted by RE'14 |
| Paper C | RQ1&2 | Plan to submit to RET'14 |
| Paper D | RQ2 | To be decided |
| Paper E | RQ3 | Published at SEAA'13 |

- **Paper A: A TASM-based Requirements Validation Approach for Safety-critical Embedded Systems Published at the 19th International Conference on Reliable Software Technologies - Ada-Europe (Ada-Europe'14), 2014.**

Requirements validation is an essential activity to carry out in the software development life cycle, and it confirms the completeness and consistency of requirements through various levels. Model-based formal methods can provide a cost-effective solution to requirements validation in a wide range of domains such as safety-critical applications. In this paper, we extend a formal language Timed Abstract State Machine (TASM) with two newly defined constructs *Event* and *Observer*, and propose a novel requirements validation approach based

on the extended TASM. Specifically, our approach can: 1) model both functional and non-functional (e.g. timing and resource consumption) requirements of the system at different levels and, 2) perform requirements validation by utilizing our developed toolset and a model checker. Finally, we demonstrate the applicability of our approach in real world usage through an industrial case study of a Brake-by-Wire system.

- **Paper B: Towards Feature-oriented Requirements Validation for Automotive Systems Accepted by the 22nd IEEE International Requirements Engineering Conference (RE'14), 2014.**

In the modern automotive industry, feature models have been widely used as a domain-specific requirements model, which can capture commonality and variability of a software product line through a set of features. Product variants can thus be configured by selecting different sets of features from the feature model. For feature-oriented requirements validation, the variability of feature sets often makes the hidden flaws such as behavioral inconsistencies of features, hardly to avoid. In this paper, we present an approach to feature-oriented requirements validation for automotive systems w.r.t both functional behaviors and non-functional properties. Our approach first starts with the behavioral specification of features and the associated requirements by following a restricted use case modeling approach, and then formalizes such specifications by using a formal yet literate language for analysis. We demonstrate the applicability of our approach through an industrial application of a vehicle locking-unlocking system.

- **Paper C: The Observer-based Technique for Requirements Validation in Embedded Real-time Systems Plan to submit to the 1st International Workshop on Requirements Engineering and Testing.**

Model-based requirements validation is an increasingly attractive approach to discovering hidden flaws in the requirements in the early phases of systems development life cycle (SDLC). The application of using traditional methods such as model checking for the validation purpose is limited by the growing complexity of embedded real-time systems (ERTS). The observer-based technique is a lightweight validation technique, which has shown its potential as a means to validate the correctness of model behaviors. In this paper, the novelty of our contribution is three-fold: 1) we formally define the observer constructs and the corresponding operational semantics and, 2) we propose the Events Monitoring Logic (EvML) to facilitate the observer specification and, 3) we show how to use observers to validate the requirements describing the system functional behaviors and non-functional properties (such as timing) of ERTS. We also illustrate the applicability of the extended TASM through an industrial application of a Vehicle Locking-Unlocking system.

- **Paper D: Title To Be Decided Conference to be decided.**

To model and analyze complicated behaviors of non-functional properties of ERTS w.r.t timing and power consumption, is a challenging yet critical task for requirements verification. We propose a solution to remedy the situation by extending TASM with the capabilities of modeling such complex system behaviors in an elegant and powerful manner using probabilistic distributions. For the purposes of requirements verification, we power up simulation-based method by using some advanced statistical techniques, with the focus on analysis of both average cases and extreme cases of the target system. More interestingly, our approach inherits the easy-to-use and easy-to-learn features from ASM, which is a literate specification language understandable and usable without extensive mathematical training.

- **Paper E: A Context-based Information Retrieval Technique for Recovering Use-Case-to-Source-Code Trace Links in Embedded Software Systems Published at the 39th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'13), 2013.**

Requirements traceability is the ability to relate requirements (e.g., use cases) forward to corresponding design documents, source code and test cases by establishing trace links. This ability is becoming ever more crucial within embedded systems development, as a critical activity of testing, verification, validation and certification. However, semi-automatic or fully-automatic generating accurate trace links remains an open research challenge, especially for legacy systems. The contribution of this paper is an improved VSM-based requirements traceability recovery approach using a novel context analysis. Specifically, the analysis method can better utilize context information extracted from use cases to discover relevant source code files. Our approach is evaluated by using three different embedded applications in the domains of industrial automation, automotive and mobile platform.

VI. TIME PLAN

The following activities, as shown in Table II, are planned to be completed before the licentiate defence. The proposed time plan starts from March, 2014:

TABLE II
TIME PLAN

| Activity | Time | Date |
|-----------------------------|---------|------------|
| Writing paper C | 5 weeks | May |
| Writing paper D | 4 weeks | June |
| First draft of the thesis | 5 weeks | July |
| Final version of the thesis | 2 weeks | August |
| Defence | | Oct |

The licentiate thesis defence can be performed in October 2014.

REFERENCES

- [1] N. G. Leveson, *Safeware: System Safety and Computers*. NY, USA: ACM, 1995.
- [2] A. T. Bahill and S. J. Henderson, "Requirements development, verification and validation exhibited in famous failures," *Syst. Eng*, 2005.
- [3] A. Ellis, "Achieving safety in complex control systems," in *Proceedings of SCSC'95*. Springer London, 1995, pp. 1–14.
- [4] M. E. C. Hull, K. Jackson, and J. Dick, Eds., *Requirements Engineering, Third Edition*. Springer, 2011.
- [5] B. Berenbach, F. Schneider, and H. Naughton, "The use of a requirements modeling language for industrial applications," in *Proceedings of RE'12*, 2012, pp. 285–290.
- [6] E. Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaitė, A. Loconsole, T. Gorscak, and R. Feldt, "Challenges and practices in aligning requirements with verification and validation: a case study of six companies," *Empirical Software Engineering*, pp. 1–47, 2013.
- [7] Model-Driven Architecture (MDA), <http://www.omg.org/mda/>, November Accessed: November 2011.
- [8] O. C. Z. Gotel and A. C. W. Finkelstein, "An Analysis of the Requirements Traceability Problem," in *Proceedings of the 2nd IEEE International Requirements Engineering Conference (RE' 94)*, 1994, pp. 94–101.
- [9] C.-H. Jane, G. Orlena, and Z. Andrea, *Software and Systems Traceability*. Springer, 2012.
- [10] N. M. A. Munassar and A. Govardhan, "A comparison between five models of software engineering," *IJCSI*, vol. 7, no. 5, pp. 94–101, 2010.
- [11] P. E. Bulychev, A. David, K. G. Larsen, M. Mikucionis, D. B. Poulsen, A. Legay, and Z. Wang, "Uppaal-smc: Statistical model checking for priced timed automata," in *QAPL*, 2012, pp. 1–16.
- [12] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "From informal requirements to property-driven formal validation," in *Proceedings of FMICS'09*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 166–181.
- [13] A. Iliasov, "Augmenting formal development with use case reasoning," in *Proceedings of Ada-Europe'12*. Springer Berlin Heidelberg, 2012, pp. 133–146.
- [14] J. Kraft, Y. Lu, C. Norström, and A. Wall, "A Metaheuristic Approach for Best Effort Timing Analysis targeting Complex Legacy Real-Time Systems," in *Proceedings of the 14th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS' 08)*, 2008.
- [15] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Trans. Softw. Eng.*, vol. 27, no. 1, pp. 58–93, Jan. 2001.
- [16] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and L. Beck, "Improving information retrieval with latent semantic indexing," in *Annual Meeting of the American Society for Info. Science* 25, 1988.
- [17] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, no. 3, pp. 993–1022, 2003.
- [18] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing supporting evidence to improve dynamic requirements traceability," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, ser. RE '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 135–144.
- [19] N. Ali, Y.-G. Gueheneuc, and G. Antoniol, "Trust-Based Requirements Traceability," in *Proceedings of the 19th International Conference on Program Comprehension (ICPC' 11)*, 2011, pp. 111–120.
- [20] W.-K. Kong and J. H. Hayes, "Proximity-based Traceability: An Empirical Validation using Ranked Retrieval and Set-based Measures," in *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE' 11)*, 2011, pp. 45–52.
- [21] A. D. Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella, "Improving ir-based traceability recovery using smoothing filters," *International Conference on Program Comprehension*, vol. 0, pp. 21–30, 2011.
- [22] A. Mahmoud, N. Niu, and S. Xu, "A semantic relatedness approach for traceability link recovery," in *Proceedings of the 20th IEEE International Conference on Program Comprehension (ICPC' 12)*, 2012, pp. 183–192.
- [23] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. Maletic, *The Grand Challenge of Traceability (v1.0)*. Springer-Verlag London Limited, 2012, pp. 343–412.
- [24] M. Shaw, "The coming-of-age of software architecture research," in *Proceedings of ICSE'01*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 656–664a.

Context-Sensitive Information Security Risk Identification and Evaluation Techniques

Dan Ionita

Services, Cybersecurity and Safety Research Group,
University of Twente,
The Netherlands
d.ionita@utwente.nl

Abstract—The objective of my research is to improve and support the process of Information security Risk Assessment by designing a scalable Risk argumentation framework for socio-digital-technical Risk. Due to the various types of IT systems, diversity of architectures and dynamic nature of Risk, there is no one-size-fits all RA method. As such, the research hopes to identify guidelines for conducting Risk Assessments in contexts that raise special challenges such as Telecom and virtualized infrastructures. Finally, it will suggest ways of qualitatively and quantitatively evaluating Information Security Risks in such scenarios by using argumentation and/or modelling attacker business cases.

I. INTRODUCTION

A Risk Assessment (RA) is a structured or semi-structured approach to analysing the security of an infrastructure, identifying weak spots, and selecting countermeasures [1]. Risk Assessments can be viewed as instances of Requirements Engineering processes as they produce desired (security) properties of the target system based on higher-level (security) goals.

Currently, (Information Security) Risk Assessments are conducted mostly via brainstorming sessions where a group of experts look at an architecture¹ and try to find and rank vulnerabilities based on the estimated likelihood and impact of their exploitation, while proposing countermeasures that could mitigate the most dangerous of these so as to reach an acceptable level of risk.

The work will be undertaken as part of the TREsPASS [2] Information Security project. The project aims to improve the way companies secure information by integrating the digital, technical and social domains with the current state-of-the-art in the field of security. One of the main goals of the project is to develop an alternative Risk Assessment methodology, built around a so-called "Attack Navigator" that can predict, prioritise and prevent common risks as well as new and complex attacks. This should provide companies with an extensive, tool-supported, automated way of conducting systematic Risk Assessments. The research described in this paper is conducted in the context of the TREsPASS project, which in turn is financed by the EU.

The project aims at being applicable to a large variety of systems, ranging from Cloud to Telecom. As such, my task is

to find out (1) which classes of targets of assessments exist and (2) which architectural views and argumentation techniques are most suitable for each one. The second goal stems from my hypothesis that, in some contexts, argumentation could compensate for the fact that probabilities and impacts are often not quantitatively known. Constructing arguments often requires referring to structural (architectural) properties of the Target of Assessment and its context. These properties can be represented using architectural views such as a communication diagram. Another useful view for arguing about certain kinds of risk could be the network business model. As such, I will also investigate the applicability of business modelling languages in defining attacker business cases and identifying and describing attacks. Finally, these results will be used to elicit requirements for the tools and techniques developed within the project.

II. MOTIVATION

Risk is commonly evaluated as the product of the likelihood and impact of an attack. Often though, these cannot be estimated quantitatively, for example because the events are very rare. As such, Risk Assessments are often forced to cope with missing or uncertain data. Even where data is available, it is mostly qualitative.

Security checklists, frequently used to support Risk Assessments, embody experience from the past and are good at detecting and mitigating known risks in an effective manner. However, problems arise when these are used on new or changing architectures or for predicting new types of risks. The rapid proliferation and development of IT systems means that architectures are constantly evolving and what used to be physical systems are now becoming IT systems too. All this raises issues with regard to the identification of new (complex) attacks in the context of a constantly evolving Risk Landscape. To make matters worse, the explosion of virtualized infrastructures (such as clouds) means that even architectures which are already in place are highly dynamic. Furthermore, using such checklists and expert judgement can pose threats to the traceability (and implicitly reproducibility) of such analyses. As such, the growing complexity and diversity of networked applications and critical infrastructures often makes checklists unusable or insufficient.

¹Throughout this paper I use the term architecture to refer to a set of roles with capabilities to influence each other so that collective behaviour emerges

I believe providing argumentation support for these (often informal) assessments, and using architectural models of the system to allow experts to reason about the risks, would facilitate not only the generation of countermeasures (or Security Requirements) but also informative justifications of these. These justifications can serve the purpose of prioritizing countermeasures and providing traceability for security requirements and investments. They can also serve as a basis for future assessments. Argumentation should cope with qualitative, missing or uncertain data by linking qualitative reasoning and quantitative computations.

Another issue is the fact that, in the context of the TReSS-PASS project, which aims at modelling entire organizations' physical, digital and social domains, factors come into play that were not relevant in traditional assessments. Besides the obvious issue of social engineering attacks, hybrid attacks are also something these methodologies are not very good at tackling. Hybrid attacks imply multiple steps which can include exploiting technical vulnerabilities, bypassing physical controls and applying social engineering techniques in a coordinated attempt to retrieve the desired information. As such, concepts like attacker skill, money flows and collaboration become more relevant and being able to identify or define "attacker business cases" might prove useful in modelling and understanding such complex attacks.

The diversity of systems targeted by Risk Assessments, from consumer applications to cloud infrastructures, means there is a wide variety of (potentially conflicting) requirements for tools and methodologies. These requirements need to be investigated and linked to particular classes of Targets of Assessment in order to provide methods that are better suited to the their application scenarios.

III. RESEARCH QUESTIONS

The central improvement problem of my research is "How can the process of identifying, analysing, evaluating and ranking relevant Risks in the field of Information Security be improved such that it works with limited or uncertain data, new/complex attacks and changing architectures in order to improve the security investment decision-making process?"

The following sub-research questions can be derived from this central improvement goal:

RQ1 What classes of Target of Assessment for Risk Assessment can be identified?

RQ1.1 Which architectural views are relevant for each class?

RQ1.2 What kind of Risk argumentation is feasible for each class?

RQ2 Design improved RA methods for these classes

RQ2.1 Which RA methods currently exist?

RQ2.2 What Risk models are implied or defined within these methodologies?

RQ2.3 What are the drawbacks of current approaches w.r.t. the various classes?

RQ2.4 What are the requirements for new methods per ToA class?

RQ3 Do the new RA methods produce the desired effects in their intended contexts?

RQ3.1 Do the new methods satisfy the requirements of each class of system?

RQ3.2 What metrics and indicators can be used to evaluate and compare the effects across the various contexts?

RQ3.3 How do the new methods compare to the state-of-the-art w.r.t the identified metrics?

IV. TECHNICAL CHALLENGES

I expect to encounter the following technical challenges during the course of this research. I will attempt to treat as many of them as possible during my research, but will focus mostly on the items at the top of the list:

- Finding the right level of formality - too much might over-encumber the analysis, decreasing usability in practice; too little would not allow for (semi-)automated reasoning, decreasing utility in practice.
- Providing traceability and justify-ability of the results - full traceability is not feasible so identifying how much traceability is actually needed in different contexts and how to achieve this sufficient level is important.
- Showing scalability of the approach - the approach has to scale well to larger Information Systems as to cope with the constant expansion of IT systems
- Pushing the boundaries of business modelling languages by using them outside of their intended scope - to describe attacker business cases, parasitic business models, illicit/unintended money flows, etc
- Integrating business modelling concepts into security models in order to reason about attacker goals and means
- Dealing with purely qualitative values and/or lack of quantitative values - as stated in the motivation, quantitative data is frequently lacking but needed for a conclusive assessment; how to work around this?
- Finding out how much of the bookkeeping and verification process can be automated thus reducing the work load of the assessors.
- Providing tool support (usability) - identifying requirements for tools that would support the process and allow for various analyses and computations via an intuitive, easy-to-use interface that can be used by non-experts.

V. STATE-OF-THE-ART

A. *Information Security Risk and Risk Assessment*

The Open Group's "Requirements for Risk Assessment Methodologies" [3] is a useful starting point as it lists high-level, empirically validated requirements that any Risk Assessment methodology should satisfy.

A literature review of current approaches [1] revealed the following potentially useful methodologies:

- FAIR [4], [5], [6], [7] methodology - One of the few methodologies that also comes with an explicit conceptual framework of Risk. Furthermore, it is part of the The Open Group's Standard for Risk Taxonomy [8])

- ISO 2700x [9], [10], [11] series of Information Security standards - currently the most important and recognized standards for Information Security; can provide useful background on current approaches.
- Structured Risk Assessment [12] methodology - unique approach to Risk Assessment, prioritizing simplicity and scalability by imposing a strict structure of the process.
- TARA [13] (Intel) methodology - claims to mitigate the issue of scalability by defining and prioritizing threat profiles; relevant due to the novel approach to describing threat classes.
- other Risk Assessment methodologies and taxonomies will also be analysed and compared.

B. Argumentation

Previous work by Toulmin [14], [15] describes inter- and intra-argument structures at a generic level. It also provides background and in-depth discussions on the uses of argument in various contexts.

The ASPIC+ argumentation framework and related work by Prakken [16] are first steps towards formal implementations of argumentation in security. However, its strict syntax raises issue w.r.t to scalability and ease-of-use.

Previous attempts at using semi-formal argumentation to justify Security Requirements mostly consist of OpenArgue [17] and related work by Haley [18], [19], [20], Franqueira and Nuseibeh [21]. While promising, these approaches are not flexible enough to handle the diversity of ToA's. Furthermore, they have not been shown scalable in practice.

C. Business Models

The e3value ontology [22], a business value modelling language, provides some of the functionality needed in order to describe Risks in Telecom scenarios. These risks are mostly misuse cases, where legitimate users make unexpected, but legal use of Telecom services in such a way that they make a profit, while causing loss for the Providers. The e3value toolkit allows modelling money and service flows and is promising in describing and identifying such Risks.

VI. SOLUTION DIRECTIONS

Given the problems listed in the Motivation section, I believe that, while a Risk Assessment meta-model can be defined, there is a need for different techniques in order to deal with different classes of Target of Assessment.

I plan to use concepts from rhetoric and argumentation to support the Risk Assessment brainstorming process in order to improve cross-expert agreement, consistency of results and traceability of countermeasures while ultimately providing semi-automated reasoning. The main goal of this research is to find a feasible mix between formal logic and informal argumentation when reasoning about Risks in networks.

In order to achieve traceability to a component level, architectural models are needed. For each class of ToAs however, some architectural views provide more useful information than others. So far I have concluded that for traditional IT systems,

diagrams of the physical architecture are most useful, but for cases like Telecom fraud, these become largely irrelevant and sometimes impossible to obtain. As such, I use business modelling ontologies to represent and analyse business cases for attackers in contexts where traditional communication diagrams are not effective. I have yet to identify a suitable architectural view for virtualised infrastructures.

I implemented an initial proof-of-concept for conducting argumentation-based RA using spreadsheets. The approach used Toulmin-style arguments in a semi-formal way to identify attack vectors and elicit security requirements for an IPTV-based system and a virtualised infrastructure. Network diagrams were to support the process. As mentioned above, initial results show applicability of such models is limited in the case of virtualised infrastructures.

I have used the e3value business modelling ontology to make business models for the attacker and defender in a couple of Telecom fraud scenarios. Except for the economic actors involved, each model shows which services or monetary transactions occur in the network in the contract period. The models also allow for spreadsheets to be generated showing break-even points for the provider and the fraudster.

VII. RESEARCH CONTRIBUTIONS

My core theoretical contribution will be knowledge about different classes of Targets of Assessment and RA requirements for each. A secondary contribution lies in exploring the capabilities of informal (Toulmin) and formal (ASPIC+) argumentation framework in analysing Information Security Risks by comparing them with established RA methodologies and using them in conjunction with various types of architectural views. Furthermore, it pushes the limits of business modelling languages, drawing conclusions on their sensitivity and extensibility while proposing new ways of using them in order to describe and hopefully identify Risks.

By extending argumentation theory with a conceptual model of Risk and identifying suitable architectural views to support this for various types of systems, I hope to improve the traceability and flexibility of Risk Assessment methodologies. As such, the practical relevance is that we will be able to do better RAs for a larger diversity of ToAs.

VIII. RESEARCH METHODS

I have conducted a literature study and review of the state-of-the art to identify current approaches (RQ2.1, RQ2.2). Case studies and Technical Action Research with the industrial partners of the TREsPASS project will be used to identify limitations of these in various contexts and requirements for improvement (RQ2.3, RQ2.4).

I will then carry out multiple iterations of Case Studies and Technical Action Research with both established and experimental RA methods in various specific scenarios (like IPTV, Cloud, Telecom) and in conjunction with various modelling languages as to identify which architectural views are more useful for each scenario (RQ1). For the initial prototypes

students will be used. Later, cross-expert evaluation will be employed to assure re-usability.

I will finally attempt to design a new set of methods and guidelines (and possibly tool) to advance the current state of the art. I will use Metric Definition Approaches (like GQM/MEDEA) to identify suitable indicators that can be used to evaluate the measure to which the goals were reached and to compare the new method(s) to established methodologies (RQ3.1, RQ3.2). Proven techniques like Gorscheck's Model for Technology Transfer [23] and Wieringa's approach to scaling up to practice [24] will be used to validate the new method(s) both within the TREsPASS project and externally via TAR(RQ3.2).

IX. PROGRESS

I started off by conducting a structured literature survey of current established Risk Assessment frameworks, methodologies, tools and standards in order to get familiar with the state-of-the-art. As indicated in the Research Methods, I also conducted several initial case studies and Risk Assessments using established methodologies in order to identify potential limitations and requirements:

- Case study and Risk Assessment of the "Follow-me" printing infrastructure of the University of Twente following the Australian/New Zealand Standard for Risk Management
- Case study and Risk Assessment of BYOD implementation in an SME based on industry guidelines (in the lack of other suitable established methodologies)
- Case study and Risk Assessment of IPTV-based home payments prototype according to the Structured Risk Assessment methodology

Initial progress towards designing a new solution so far includes:

- Experiment in using formal logic framework (ASPIC+) to support Risk Assessment with Prakken[16]
- Spreadsheet-based tool and proof-of-concept aiming at a middle ground between Prakken's formal approach [16] and OpenArgue's less formal method [17].
- Applying the e3value ontology to model business cases for malicious users in several Telecom fraud scenarios.

The following were carried out in order to validate the initial solutions:

- Case Study and Risk Assessment of generic IaaS Cloud infrastructure with IBM Research Zurich using the spreadsheet-based tool and public Cloud Risk knowledge bases [25], [26]
- Multiple case studies and Risk Assessments of IPTV home-payments system using the spreadsheet-based tool
- Collaboration with the Deutsche Telekom Chair of Mobile Business & Multilateral Security for validating the e3value approach in Telecom misuse scenarios

X. ACCEPTED PUBLICATIONS

- H. Prakken, D. Ionita, and R. Wieringa, "Risk assessment as an argumentation game" in *Computational Logic in*

Multi-Agent Systems. Springer Berlin Heidelberg, 2013, pp. 357373.

REFERENCES

- [1] D. Ionita, *Current established risk assessment methodologies and tools*, July 2013. [Online]. Available: <http://essay.utwente.nl/63830/>
- [2] "Technology-supported risk estimation by predictive assessment of socio-technical security," 2012-2016. [Online]. Available: <https://www.trespass-project.eu/>
- [3] T. O. Group. (2009, January) Requirements for Risk Assessment Methodologies.
- [4] R. M. I. LLC, *FAIR (FACTOR ANALYSIS OF INFORMATION RISK) Basic Risk Assessment Guide*. Risk Management Insight LLC, 2006.
- [5] T. O. Group, *Technical Guide: Fair - ISO/IEC 27005 Cookbook*, October 2010, no. C103.
- [6] R. M. I. LLC, *FAIRLite High-Level Description*. Risk Management Insight LLC, 2010.
- [7] J. A. Jones, "An Introduction to Factor Analysis of Information Risk (FAIR)," Risk Management Insight, 2005.
- [8] T. O. Group, *Technical Standard to Risk Taxonomy*, January 2009, no. C081.
- [9] "Information technology – Security techniques – Information security management systems – Requirements," 2005.
- [10] "Information technology – Security techniques – Code of practice for information security management," 2005.
- [11] "Information technology – Security techniques – Information security risk management," 2011.
- [12] N. A. McEvoy and A. Whitcombe, "Structured risk analysis," in *Proceedings of the International Conference on Infrastructure Security*, ser. InfraSec '02. London, UK, UK: Springer-Verlag, 2002, pp. 88–103.
- [13] M. Rosenquist, "Prioritizing information security risks with threat agent risk assessment," "http://www.communities.intel.com/servlet/JiveServlet/download/4693-1-3205/Prioritizing_Info_Security_Risks_with_TARA.pdf", December 2009.
- [14] S. E. Toulmin, *The Uses of Argument*. Cambridge University Press, July 2003.
- [15] S. Toulmin, R. Rieke, and A. Janik, *An introduction to reasoning*. Macmillan, 1979.
- [16] H. Prakken, D. Ionita, and R. Wieringa, "Risk assessment as an argumentation game," in *CLIMA*, ser. Lecture Notes in Computer Science, J. Leite, T. C. Son, P. Torroni, L. van der Torre, and S. Woltran, Eds., vol. 8143. Springer, 2013, pp. 357–373.
- [17] Y. Yu, T. T. Tun, A. Tedeschi, V. N. L. Franqueira, and B. Nuseibeh, "Openargue: Supporting argumentation to evolve secure software systems," in *RE*. IEEE, pp. 351–352.
- [18] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Trans. Softw. Eng.*, vol. 34, no. 1, pp. 133–153, Jan. 2008.
- [19] C. B. Haley, R. C. Laney, B. Nuseibeh, W. Hall, C. B. Haley, R. C. Laney, and B. Nuseibeh, "Validating security requirements using structured toulmin-style argumentation," 2005.
- [20] R. M. J. D. Haley, Charles B; Laney and B. . Nuseibeh, "Arguing satisfaction of security requirements."
- [21] V. N. L. Franqueira, T. T. Tun, Y. Yu, R. Wieringa, and B. Nuseibeh, "Risk and argument: A risk-based argumentation method for practical security," in *RE*. IEEE, 2011, pp. 239–248.
- [22] J. Gordijn and H. Akkermans, "Value based requirements engineering: Exploring innovative e-commerce idea," *Requirements Engineering Journal*, vol. 8, no. 2, pp. 114–134, 2003.
- [23] T. Gorscheck, P. Garre, S. Larsson, and C. Wohlin, "A model for technology transfer in practice," *IEEE Software*, vol. 23, no. 6, pp. 88–95, 2006.
- [24] R. J. Wieringa, "Empirical research methods for technology validation: Scaling up to practice," *Journal of systems and software*, vol. online pre-publication, 2014.
- [25] "ENISA: Cloud Computing Security Risk Assessment," May 2009. [Online]. Available: http://www.enisa.europa.eu/pages/02_03_news_2009_05_18_risk_survey.html
- [26] "Top Threats to Cloud Computing V1.0," 2010.

Business Processes and Regulations Compliance Management Technology

Ilze Buksa

Institute of Applied Computer Systems

Faculty of Computer Science and Information Technology, Riga Technical University

Meza Street 1/4-547, Riga LV-1048, Latvia

Ilze.Buksa@rtu.lv

Abstract—Organizations must comply with a number of external and internal regulations for business governance and must ensure that their processes are running accordingly to requirements of regulations. Therefore business process and regulations compliance analysis audit and management tasks take a very important role in daily operational activities for each organization. Due to high complexity this task can become challenging. In some domains regulations are changing rapidly. Process models must be flexible and easy adjustable to changing requirements, since the lack of ability to comply with regulations can lower down the competitiveness of an organization. Thus, rapid changes of regulations require rapid changes of related business processes. The goal of the PhD work is to develop business process and regulations compliance management technology which should enable business process construction from regulations and change monitoring of regulations and business processes to ensure easy and rapid modification of business process model or regulations, thus also ensuring compliance.

Index Terms—business process, regulations, compliance.

I. RESEARCH QUESTIONS AND HYPOTHESES

In most organizations business processes are affected by requirements of regulations. Many organizations use business process management methodologies to model, simulate, execute, monitor and optimize business processes. Business process models can also serve as a proof that organization is running accordingly to regulations. To ensure such a proof, organizations must form and manage linkage between regulations and business processes [1].

With the regulations the author means documents which record knowledge about “what” and “how” the goals of the organizations must be achieved and products or services produced or provided [9], in other words, any document which contain valid source of regulatory requirements according to which business processes of the organization must be executed [1].

From an organization's point of view, the challenge is the traditional approach of treating regulations separately from business processes [2]. At a high level of abstraction regulations can be divided into the following categories: mandatory regulations, which are issued by governing bodies; “good to have” non-mandatory regulations such as various industry standards; and internal regulations, which are

developed by enterprise. From the enterprise point of view, the first two types of regulations are regarded as external regulations. Internal regulation may depend on (or mirror) external regulations as well as they may be independent of external regulations [2].

Usually organizations ensure business process compliance with regulations through audits. Due to the need to provide regular updates on compliance, a more strategic approach of compliance management is needed. This implied a shift from regular reviews to continual assurance and introduced a need for advanced compliance management methodologies and tools that reflect and ensure real time compliance management [2].

In many cases organizations have their business processes specified independently or not specifically linked with regulations, even if business processes are directly impacted by regulations, e.g. in governmental and municipal organizations. Reasons for such situation can be lack of widely known and easily applicable methodologies and weak support of easy to use IT tools to link business processes to regulations [4].

Since requirements under which an organization must operate are expressed in regulations and business processes represent procedure how these requirements should be executed, it is a rational need to find a way to link these two concepts. Characteristics of regulations make them challenging to use and directly apply in business processes, therefore captured (identified) business process elements within regulatory text can be a point to link regulatory documents with business process model.

The following issues are selected for further investigations as research questions in order to address business process and regulations compliance management [4]:

- *Management of regulations* - extraction of regulatory requirements; development of regulatory document generation and update algorithm, solutions for versioning of regulations.
- *Linkage of regulations to business processes* - solutions to link business processes with regulations and representation capabilities of regulatory text parts/elements within business process model and business process elements.

- *Compliance management* - compliance algorithms and software tools capabilities to enable regulations and business process compliance management.

Goal of the PhD work is to develop business process and regulations compliance management technology which will consist of the method & tool prototype. Developed method & tool should address and provide solutions for the following topics:

- Management of regulations - solutions for retrieval, storing, updating, and versioning of regulations).
- Extraction of requirements from regulations - solution for capturing requirements from regulations which are affecting business processes.
- Linkage of business processes and regulations.
- Business process and regulations mutual change management and on-time compliance monitoring.

II. THE STATE OF THE ART

There is a lot of research available that propose approaches and solutions for analyzing regulations, capturing requirements from regulations, facilitating understanding of regulations by graphical representation offering various modelling languages and syntaxes, and managing compliance with requirements and business processes.

Kharbili [10] presents a conceptual framework and thorough analysis of existing approaches to automated regulatory compliance management. Relationships between business processes and regulations are analyzed and some suggestions about their monitoring are provided in research [11]-[15].

The most important work that is related to my PhD work are the following:

Kiyavitskaya et al [7] have proposed method for extracting rights and obligations from regulations and continuing the research [8] they have examined tools to support this by applying Cerno framework for textual semantic annotation and have proposed a tool for semi-automatic semantic annotation of concepts. In research authors have focused on alignment of information system requirements with regulations. However, ideas of research can be analyzed and expanded with focus to align requirements with business processes. Also national (in this case Latvian) language semantic challenges need to be addressed in further research of PhD work.

Araujo et al [4] represent the method for validating business processes with respect to business rules that are captured from regulations. The method does not address challenges of extracting requirements from regulations; however it is practical and applicable approach to trace and visualize compliance.

Governatori et al [6] validates business processes with business contracts by providing logic-based formalism for describing both semantics of contract and semantics of compliance checking procedures. Approach is valid in scope of PhD research and will be applied in further case studies by testing it to regulations, as both, regulations and contracts, contain requirements for business processes.

Also other research [21]-[28] ideas might be applicable in scope of further research.

Process management and governance frameworks are under investigation as they specify how processes can be managed or governed. Specific regulations and governance frameworks include, but are not limited to:

- Basel II [16] - regulation for the banking and financial sectors, which is the required implementation of the EU in all EU countries since January 1, 2007.
- Sarbanes-Oxley Act (SOX) [19] - federal law in the United States of America to ensure the accuracy and reliability of published financial data.
- Control Objectives for Information and Related Technology (COBIT) [17] - comprises a range of the best practices for IT management and control. COBIT is a top-down approach from the enterprise goals, to the derived IT goals, to their impact on IT architecture [18].

III. AUTHOR'S CONTRIBUTIONS AND ORIGINALITY

Several research topics have been covered so far in scope of PhD work with great contribution of supervisor professor and in collaboration with other colleagues.

This sections further details research made so far [1], [2], [3].

In research paper [1] a high-level architecture of Document Analysis & Change detection system has been proposed, responsible for retrieval of the regulations, document analysis and preparation for linkage to the business processes.

To enable linkage of the business process to the regulations in the business process management tools, authors of research faced the following challenges: (1) analysis of regulations, (2) detection of changes in regulations and application of detected changes to related business process models and elements, and (3) linkage integration in the business process management tools [1].

The goal of the research was to describe an approach to transform regulations into the form that contains annotated structural parts such as chapters, sections, articles, and sub articles that facilitate business process linkage to the specific structural part of relevant regulation, and link these structural parts to the business process. Also an approach for business process and regulations' change management was proposed. By applying this approach of regulations' change management authors could ensure regulation change monitoring thus facilitate up-to-date business process compliance with the regulations [1].

Results of the approach [1] were tested in practical example in ARIS business process management tool. Basic concept of the proposed approach are that regulation can be annotated and divided into structural parts thus enabling linkage options from particular business process activities to specific parts of the regulation. For annotation of regulation, UIMA annotation was selected. UIMA is appropriate for identification of structural parts of the regulation, because it ensures simple development and usage mechanism of the new components (Analysis Engine, UIMA PEAR). However the developed Paragraph

annotator should be improved to ensure higher precision of the detection of the structural parts of the regulation [1].

Research [2] reports an experience of authors in creating an enterprise model compliant with the Latvian Accounting Law. The focus is on a possibility to represent parts of the law in the form of business processes. The issues that the law considers together with the information on processes are organized in related sub-models. The main elements of the enterprise model sufficient for representing issues prescribed by the regulations were presented and discussed. The suitability of the de facto business process modeling standard BPMN 2.0 for representing regulations were examined [2]. The paper reports on enterprise modeling experiment that is based on a representation of regulations as reusable business process model parts. The experiment showed that for proper positioning of the parts it is necessary to represent in models not only the process as sequence of elements, but also other related information available in regulations. The paper proposes the enterprise model suitable for modeling regulation. The comparison of this model to a well-known enterprise model helps to see that the enterprise model has to include an events model as one of its sub-models for regulations modeling purposes [2].

The paper [2] contributes with described and illustrated limitations of BPMN 2.0 in its applicability for regulations modeling. It is a matter of future research to overcome these limitations, since due to its popularity the BPMN is still the main candidate for modeling regulations in situations where models are developed for public use [2]. Further experiments with other regulations may reveal some new requirements for enterprise and process models. The general aim of the research [2] is to provide reusable business process model parts (that mirror regulations) in cloud solution [3] in order to enable easier enterprise business process compliance to regulations.

In Research [3] authors proposes an approach for managing regulation-dependent business process parts in cloud solution. The commonalities and differences between business processes of organizations depend on the level of abstraction at which the processes are represented [3]. At lower levels of abstraction the processes are complex and their variations cannot be captured by specific business process frameworks. However there are some parts of processes that are common to many enterprises or several units in an enterprise, for instance, the process parts that must conform to particular regulations [3]. The purpose of the approach is to minimize the total time, which organizations use for incorporation of regulations in their business processes [3].

Cloud solution considers the following capabilities [3]:

- Enterprise can access its private resources such as internal regulations stored on the cloud.
- Multiple enterprises can access the same resources such as external regulations [3].

Enterprise business processes must comply with or take into consideration many different regulations. Parts of these regulations are common to several organizations or several processes in the same organization. Analysis of these regulations and their inclusion in business process models require large amount of time and effort. In this paper authors

envision an approach where regulations are translated into business process model “spare parts” or raw materials that can be used by designers of business processes at several enterprises (or several units in one enterprise) [3]. The solution is based on the use of the regulation digraph, which is related to the regulations and the business process “spare parts” or raw materials amalgamated in the spare parts repository. The research presented in the paper only blueprints the approach and needs further research and experiments [3].

IV. THE RESEARCH METHODS

Primary research methods are collation, summary and synthesis of existing research and information in scope of dissertation thesis, as well as original research to develop solutions for the identified problems through creating and validation of new approaches using design science approach.

Also an evaluation of a prototype to provide credible evidence of results in target user groups, collection and analysis of user feedback and evaluation of feedback against the goals of the research to be achieved is planned.

V. A DESCRIPTION OF THE PROGRESS

A. Current Progress of the Research

There were the following stages set to reach goals of PhD work:

- (1) To collect, analyze and to get introduced with background materials and related publications.
- (2) To create method to determine applicable regulations to business processes; to classify and structure regulations; to ensure traceability within regulation itself and other applicable regulations.
- (3) To create method to extract (capture) requirements from applicable regulations; transform or link regulatory constructs to business process elements and models.
- (4) To link business process model and steps with regulatory document; to model business processes incorporating requirements/regulatory constructs; to handle inconsistencies between regulatory requirements.
- (5) To ensure traceability between business process and its steps with requirements, to ensure that all applicable requirements are present.
- (6) To provide real time compliance monitoring and change detection; to create change detection procedure and solution architecture.
- (7) To develop software solution prototype.

From above list following tasks have already been addressed: task (1) is in constant progress to follow news in research themes associated with PhD work; tasks (2) – (3) are partly done in research [1] and [3], however additional research and validation of new ideas are required; tasks (4) – (5) are also partly done in research [1], [2] and [3], but also additional research is required; tasks (6)-(7) yet to be performed. However, main activities to combine work done so far in

applicable method and conduct further research to reach the goals are still to be completed.

B. Further Research

Develop method to identify business process elements in regulatory text (manually or automatically); analyze and select suitable business process modelling language; develop compliance monitoring method and algorithm; validate solution idea with target user groups and develop software solution prototype.

ACKNOWLEDGEMENT

The research has received funding from the research project "Information and Communication Technology Competence Center" co-financed by European Regional Development Fund contract nr. L-KC-11-0003, signed between ICT Competence Centre and Investment and Development Agency of Latvia, Research No. 1.13 "Research for automated analysis of normative documents and business process compliance management", implemented by JSC "RIX Technologies".

ACCEPTED PUBLICATIONS

- [1] P. Rudzājs and I. Bukša "Business Process and Regulations: Approach to Linkage and Change Management". In: 10th International Conference on Perspectives in Business Informatics Research: 10th International Conference on Perspectives in Business Informatics Research, Latvia, Riga, 6-8 October, 2011. Springer: Springer, 2011, pp.96-109. ISBN 9783642245107.
- [2] L. Būšinska, M. Kirikova, L. Peņicina, I. Bukša, and P. Rudzājs "Enterprise Modeling for Respecting Regulations" In: PoEM Short Papers 2012 "Emerging Topics in the Practice of Enterprise Modelling", Germany, Rostock, 7-8 November, 2012. Rostock: CEUR, 2012, pp.106-118. ISSN 1613-0073.
- [3] M. Kirikova, I. Bukša, L. Peņicina, "Raw Materials for Business Processes in Cloud" In: Enterprise, Business-Process and Information Systems Modeling: 13th International Conference BPMDS 2012: Conference Proceedings, Poland, Gdansk, 25-26 June, 2012. Berlin: Springer Berlin Heidelberg, 2012, pp.241-254. ISBN 9783642310713. e-ISBN 9783642310720. ISSN 1865-1348.
- [4] I. Buksa "Business Process and Regulations Compliance Management Technology", CAiSE (Doctoral Consortium) 2011:1.

REFERENCES

- [5] B. M. Araujo, E. A. Schmitz, A. L. Correa, and A. J. Alencar "A method for Validating the Compliance of Business Processes to business Rules". Proceedings of the ACM Symposium on Applied Computing SAC'10, 2010.
- [6] T. Eijndhoven, M. E. Iacob and M. L. Ponisio "Achieving business process flexibility with business rules". Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference EDOC'08, 2008.
- [7] G. Governatori, Z. Milosevic and S. Sadiq "Compliance checking between business processes and business contracts". Proceedings of the 10th International IEEE Enterprise Distributed Object Computing Conference EDOC'06. p.221-232, 2006.
- [8] N. Kiyavitskaya, N. Zeni, T. D. Breaux, and A. I. Anton "Extracting Rights and Obligations from Regulations: Toward a Tool-Supported Process". Proceedings of the 22nd International IEEE/ACM conference on Automated software engineering ASE'07. p.429-432, 2007.
- [9] N. Kiyavitskaya, N. Zeni, T. D. Breaux, A. I. Anton, J. R. Cordy, L. Mich, and J. Mylopoulos "Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations". Proceedings of the 14th International IEEE Requirements Engineering Conference RE'06. p.49-58, 2006
- [10] M. Kirikova "Facilitating Comprehension of Normative Documents by Graphical Representations. Practical Aspects of Knowledge Management" Springer Verlag, Berlin Heidelberg. p.369-376, 2002
- [11] M. E. Kharbili "Business process regulatory compliance management solution frameworks: a comparative evaluation". The Eighth Asia-Pacific Conference on Conceptual Modelling (APCCM), 2012.
- [12] S. Olbrich and C. Simon "Integration of legal constraints into business process models", Transforming Government, People, Process and Policy, Emerald Group Publishing Ltd., Vol. 1, No. 2, pp 194-210, 2007.
- [13] G. J. Veal and St. Mouzas "Changing the rules of the game: Business responses to new regulation", Industrial Marketing Management, Elsevier, Vol. 40, Issue 2, pp 290-300, (2011)
- [14] A. Boer and T. van Engers "AGILE: From source of law to business process", Proceedings of the Third Workshop on Legal Ontologies and Artificial Intelligence Techniques, held in conjunction with the Workshop on Semantic Processing of Legal Texts. Barcelona, Spain, June 8, 2009, N. Casellas, E.C. Francesconi, R. Hoekstra, and S. Montemagni (Eds.), available at <http://ceur-ws.org/Vol-465/>
- [15] G. Governori "Law, Logic and business processes", Proceedings of the 2010 Third International Workshop on Requirements Engineering and Law (RELAW), pp. 1-10, 2010
- [16] A. Awad, M. Weidlich and M. Weske "Visually specifying compliance rules and explaining their violations for business processes", Journal of Visual Languages and Computing Vol. 22, Elsevier, pp. 30-55, 2011.
- [17] Basel Committee on Banking Supervision, Basel II Accord, <http://www.bis.org/publ/bcbcsa.htm>
- [18] Control Objectives for Information and Related Technology (COBIT), <http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx>
- [19] T. Weilkiens "OCEB Certification Guide: Business Process Management - Fundamental Level", The MK/OMG Press, 2011
- [20] Sarbanes-Oxley Act of 2002, Public Law 107-204 (116 Statute 745), United States Senate and House of Representatives in Congress, <http://www.soxlaw.com/>
- [21] J. Slankas and L. Williams, "Automated Extraction of Non-functional Requirements in Available Documentation," in International Conference on Software Engineering (ICSE) 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), pp. 9-16, 2013.
- [22] T. D. Breaux and A. I. Antón, "Analyzing Regulatory Rules for Privacy and Security Requirements" IEEE Transactions on Software Engineering, vol. 34, pp. 5-20, 2008.
- [23] Casamayor, D. Godoy and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," Information and Software Technology, vol. 52, pp. 436-445, 2010.

- [24] V. Ambriola and V. Gervasi “Processing natural language requirements”. In Proceedings of the 12th international conference on Automated software engineering, pp. 36-45, 1997.
- [25] L. Goldin and D. M. Berry “Abstfinder, a prototype abstraction finder for natural language text for use in requirements elicitation: design, methodology, and evaluation”. In Proceedings of the First International Conference on Requirements Engineering, pp. 18–22, 1994.
- [26] B. S. Lee and B. R. Bryant “Automation of software system development using natural language processing and two-level grammar” Radical Innovations of Software and Systems Engineering in the Future. Springer, Heidelberg pp. 219–233, 2004.
- [27] S. Ingolfo, A. Siena, I. Jureta, A. Susi, A. Perini and J. Mylopoulos “Choosing Compliance Solutions through Stakeholder Preferences” REFSQ, pp. 206-220, 2013.
- [28] A. Siena, S. Ingolfo, A. Perini, A. Susi and J. Mylopoulos “Automated Reasoning for Regulatory Compliance” ER, pp. 47-60, 2013.

Creative Strategic Scenarios for Preparation to Requirements Evolution

Marília Guterres Ferreira

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

Rio de Janeiro - RJ, Brazil

mferreira@inf.puc-rio.br

Abstract—The focus of this research is Creative Strategic Scenarios as predictive models of software evolution for socio-technical systems in organizations. This research seeks to combine theories of Strategic Planning and Creativity to generate strategic scenarios that could predict Organizational Changes. The work will integrate scenarios and the i^* goal modelling mechanism to analyse the impacts of organizational change through strategic scenarios.

Index Terms—Requirements Engineering, Software Evolution, Strategic Planning, Scenario Planning, Creativity.

I. INTRODUCTION

An organization can be viewed as a set of individual efforts with the purpose of achieving collective purposes and it is connected to an external environment. This social organization is supported by organizational information systems, which in turn are supported by software systems. To keep their competitiveness, organizations evolve and organizational evolution is motivated by internal and external factors. As an example of internal factor, changes to personnel turnover or retirement, this will require changes in the organization, driving changes to the software as well. As an example of external factor, the organization is inserted into a constantly changing marketplace, and its changes may lead to organizational changes. In this context, strategic planning comes to support this dynamism and to achieve the company's goals. To contribute to this purpose, strategic scenarios can be the input of these activities. Furthermore, these strategic decisions need to be creative to anticipate and to pre-emptively prepare the organization and its software systems for organizational changes.

Organizations should keep evolving to maintain their competitiveness in the marketplace. They have to respond quickly to changes of the market, of their clients' needs or of organization goals [1]. The organization sustainability depends on anticipating and reacting to sequences of voluntary and emerging change [2]. In this setting, Software Evolution is currently done under time pressure and some changes can be anticipated in order to better adapt *Software Systems that support Organizational Information Systems (SSsOIS)* [3]. We will focus on the Lehman's Laws of Software Evolution that relates the *Continuing Change* of E-systems to their *Increasing Complexity*, *Continuing Growth* and *Declining Quality*, mostly

laws' numbers I, II, VI, VII and VIII [4][5], as well as on the lack of accurate Predictive Models in Software Evolution [6].

To help the anticipation of organizational changes related to the evolution of SSsOIS, strategic planning can be seen as a creative process in which organizational futurologists work together to create ideas for new strategic scenarios that are eventually expressed as i^* models [7]. Therefore, as strategic planning can be framed as a creative problem solving, strategic practitioners can recruit relevant theories, models, techniques and tools from creative problem solving to understand and support requirements processes more effectively [8] [9].

This Doctoral Research is centred on the hypothesis of contributions from visions of organizational evolution and strategic planning with an emphasis on creativity to generate Creative Strategic Scenarios (CSS) as predictive models for software evolution to support organizational information systems. This Doctoral Research addresses SSsOIS and software represented by “*Conceptual Models that support Requirements Engineering (CMsRE)*”.

II. PROBLEM STATEMENT

Handling Changes is considered one of the major challenges the community is facing [10]. Namely, even when the organization and the SSsOIS are aligned, requirements are going to evolve due to organizational changes demanded by the dynamicity of the organizational environment. Because of this, over time, software systems present inconsistencies and lack of compliance with new environmental requirements in which they were deployed.

III. AIM AND OBJECTIVES

The main aim of this Doctoral Research is to study how to adapt SSsOIS to organizational changes pre-emptively.

From this, the objectives are:

- 1) Study how to predict organizational changes;
- 2) Study how to identify impacts of organizational changes in CMsRE;
- 3) Study how to identify points of change on CMsRE.

IV. ASSUMPTIONS

To achieve the objectives of this Doctoral Research, some assumptions were adopted:

- 1) If organizational changes could be anticipated, a CMsRE could be adapted pre-emptively;
- 2) Strategic decision have impacts of broad spectrum in the organization;
- 3) Writing conceptual models having different purposes in the same language helps identify impacts between them.

V. RESEARCH QUESTIONS

This Doctoral Research is driven by the following Research Questions:

- 1) How to anticipate organizational changes?
- 2) How to identify impacts of organizational changes in CMsRE pre-emptively?
- 3) How to adapt a CMsRE to organizational changes pre-emptively?

VI. RESEARCH HYPOTHESES

The general Hypothesis of this Doctoral Research is:

“The use of CSS provides basis for predicting changes at SSsOIS”.

Auxiliary hypotheses are exposed below:

- 1) The following areas can contribute to the construction of CSS:
 - a) Strategic Planning
 - b) Organizational Evolution;
 - c) Organizational Change;
 - d) Organizational Futurology;
- 2) Creativity theories can assist the construction of CSS;
- 3) A Conceptual Model CSS is compatible with a CMsRE.
- 4) It is possible to write both a CSS and a CMsRE at the same modelling language.

VII. STRATEGY TO BE INVESTIGATED

This Doctoral Research comes to a preliminary strategy to be investigated in order to address the problem stated and to achieve the goals exposed. It presented as follows:

- 1) Construction of Creative Strategic Scenarios to support the anticipation of Organizational Changes:
 - a) Basis:
 - i) Organizational Evolution;
 - ii) Organizational Change;
 - iii) Organizational Futurology;
 - iv) Creativity;
 - b) Representation Languages for modelling CSS and CMsRE:
 - i) Language that is understood by organizational futurists;

ii) Language Extended Lexicon (LEL) [11];

iii) Scenarios [12] [13];

iv) i* [14];

- 2) Impact Analysis between CSS and CMsRE to identify the impacts of organizational changes anticipated;
- 3) Identification of points of changes in CMsRE in order to adapt them to organizational changes pre-emptively.

VIII. RESEARCH PROPOSAL

This Doctoral Proposal is presented as an activity model using SADT (*Structured Analysis and Design Techniques*). This technique uses a hierarchical decomposition of activities with a top-down approach, wherein each new diagram, the activities of the previous level are decomposed into three to six others. Moreover, the boxes represent activities, the left incoming arrows are inputs of these activities, the right outgoing arrows are outputs, the bottom incoming arrows are the means, components or tools used and the upper incoming arrows are the controls that influence the execution of the activity [15]. Fig. 1 depicts Level 0 of this Doctoral Proposal which is: *Prepare Requirements for Software Evolution based on Alternatives Futures oriented by Strategic Planning and Strategic Thinking theories using Creativity*.

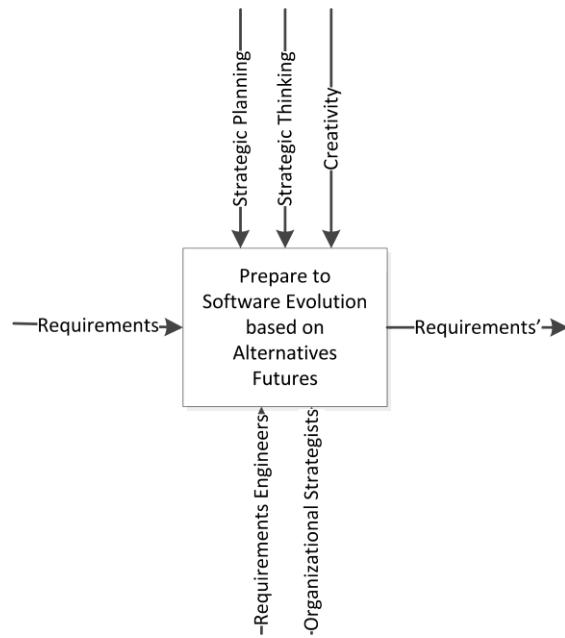


Fig. 1. Level 0 – Proposal

Then, this objective is decomposed in three activities that represent this Research Questions, as showed at Fig. 2.

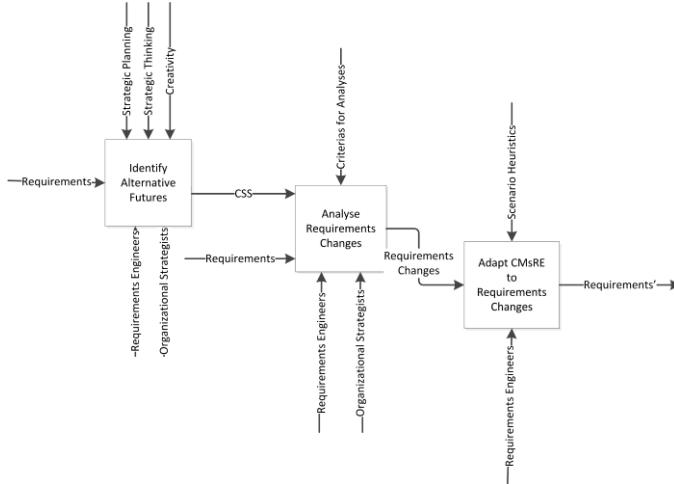


Fig. 2. Level 1 - Prepare to Software Evolution based on Alternative Futures

In order to *Identify Alternative Futures*, we apply Scenario Planning as a tool to construct scenarios of possible futures. *Strategic Planning* and *Strategic Thinking* theories will guide this construction as well as Creativity Framework. The output of these activities will be validated CSS of possible futures aligned to the Organizational Aspects.

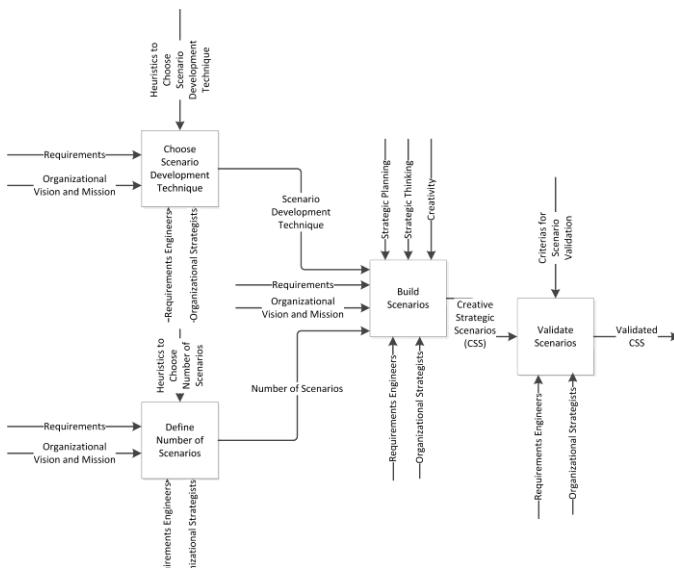


Fig. 3. Level 2 – Identify Alternative Futures

Based on the CSS, next activities aims to identify how the requirements should behave in these alternative futures. Requirements and CSS will be analysed in order to point the Requirements Changes, its risks, impacts and prioritization. The definition of requirements changes will be the output of these activities.

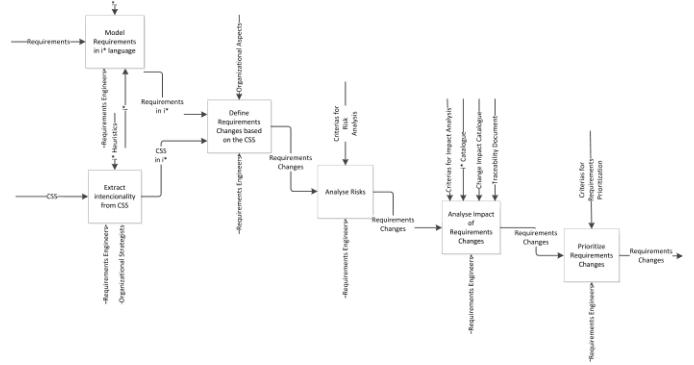


Fig. 4. Level 2 – Analyze Requirements Changes

Next step is to pre-emptively adapt the CMsRE to the Requirements Changes. In this phase, the CMsRE, here represented by scenarios, will be evolved aligned to the Requirements Changes.

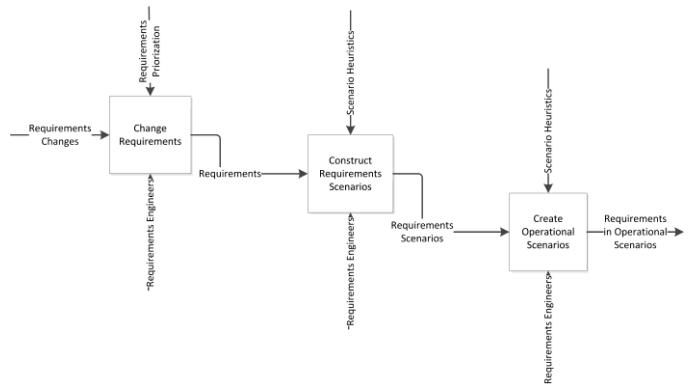


Fig. 5. Level 2 – Adapt CMsRE to Requirements Changes

Fig. 6 summarizes the relationship between the concepts approached by this Doctoral Research and the objective of their relationships.

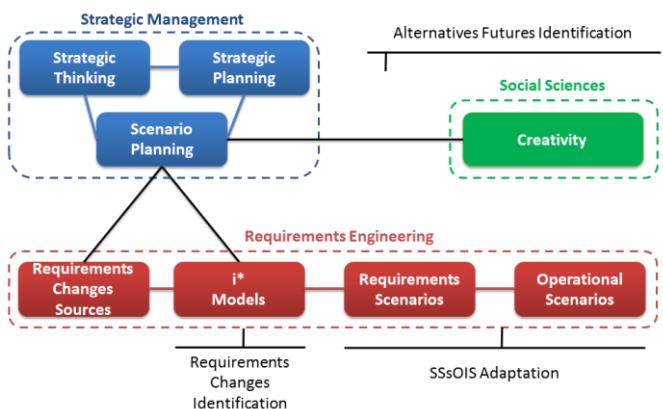


Fig. 6. Concepts approached in this Research

IX. STATE OF THE ART

The works listed below are the foundation of this research and are briefly examined as follows:

Macedo's Doctoral Thesis worked with Continuous Organizational Change and how to strategically deal with it using Knowledge Management support [16]. It proposes an architecture for corporate memory with emphasis on strategic issues, hence the name Strategic Corporate Memory - SCM, which takes into account the competitive situation of the global market. The architecture of SCM reflects their Organizational Baseline, which is the repository of knowledge that aims to meet this range of information needs, based on the business conceptual model. The modelling (abstraction) of the business model is conducted taking as inspiration the different approaches to strategic analysis, notably the visions of "positioning" and "emphasizing efficiency", therefore, using an integrated view of these proposals and enriched with an analysis of the processes and functions under a total quality orientation and metaphors about the images of the organization. This work substantially contributes to this Doctoral Research concerning to the Organizational Aspects, as Strategic Planning and Continuous Change, and relating them to the modelling of Software Systems.

Breitman and Leite developed an extensive research on Scenario Evolution, from elicitation to coding [17] [18] [19] [20]. They work on the Evolution of Scenarios due to the Learning Process intrinsic to Requirements Engineering, presented in [21] and in [22]. In their work, they address the lack of a specific methodological support to the use of scenarios. From this problem, they aim to understand what is required for the effective use of scenarios in system development, and propose an organization that supports an engineering approach to scenario evolution. The organization is based on domain knowledge embodied in the scenario evolution model and takes into consideration other software management issues, such as configuration management and traceability. Their research presents aspects co-related to this with respect to the use of scenarios. Therefore, it provides methodological support for the use of scenarios which is very important for this work.

Colette Rolland has been publishing insightful papers in the area of Requirements to Software Evolution for more than 20 years [23]. In the year of 2006, Rolland, Salinesi and Etien came up with an Alignment and co-evolution Method (ACEM) [1]. They investigated the alignment between IT systems and the business they support; and also how system evolves due to contextual forces. For the former, Rolland takes the position that intentional modelling can help resolving some of these issues. For the later, they propose a generic typology of gaps to facilitate a precise definition of change requirements, by means of modelling change as a set of gaps between the requirements specification of the current and the future system. Their work provides solid foundation for the co-evolution approach with regard to the interdependence between the organization and the IT system. Their contributions will guide how to apply intentional modelling to represent the organizational factors to IT systems.

Ernst, Mylopoulos and Borgida examined the issue of Software Evolution from a Requirements Engineering perspective [24] [3]. In Ernst's Ph.D. dissertation, they followed the Requirements Problem approach, wherein software development can be characterized as finding a specification that satisfies user requirements, subject to domain constraints. To enable this, they proposed a shift from treating requirements as artefacts to treating requirements as design knowledge, embedded in knowledge bases. They introduced a Requirements Engineering Knowledge Base (REKB) and the main result of their work on the REKB is a tool and approach which can guide software developers and software maintainers in design and decision-making in the context of software evolution. Their work provides guidelines to relate Requirements Engineering with Software Evolution. Besides that, their results can also be either extended or compared to those achieved by this work.

Pumareja and Wieringa studied the relationship between socio-technical systems and Software Evolution [25] [26]. In her Ph.D. dissertation, they analysed the evolution of requirements in groupware systems (applications that support the cooperative process of individuals working as a group). Through empirical investigation by means of case studies, their most important contribution is a set of requirements evolution patterns. Pumareja's work is helpful to this research with respect to the domain studied, the research method employed and the results attained. First, they explored socio-technical systems that can bring about profound organizational change based on studies that point to emergent organizational properties; this can direct theoretically this research in the subject of organizational aspects related to Software Evolution. Second, they applied a conceptual framework of Software Evolution to study real cases in this context; this method can guide the studies to be made in this work. Finally, their results can be used either for extension as for comparison to those achieved by this work.

Bryl and Giorgini researched the growing involvement of humans and organizations in system structure and operation [27] [28]. They argue that an interdisciplinary notion of a Socio-Technical System (STS) is the one that captures aspects as the organizational environment in which software operates, the software system itself, the related hardware components and human users. They address the problem of understanding the requirements of the STS software component and the way in which the structure of human and organizational activities is influenced by introducing technology. Then, they present a framework, which aims at supporting the design of STS, specifically the design of a network of inter-actor dependencies intended to fulfil a set of initial goals. The focus on the relationship between socio and technological aspects in the design of socio-technical systems is the one applied in this Doctoral Research. However, they propose Requirements Engineering specifically to socio-technical systems and this Doctoral Research proposes Requirements Engineering to evolve socio-technical systems. Their results are very valuable and relevant for this Doctoral Research.

X. RESEARCH METHOD

The Research Method applied in this Doctoral Research is guided by the Research Framework, proposed by Wieringa, Maiden, Mead and Rolland, illustrated in Figure 1 [29]. It is called Engineering Cycle and its activities will be followed in the development of this Doctoral Research.

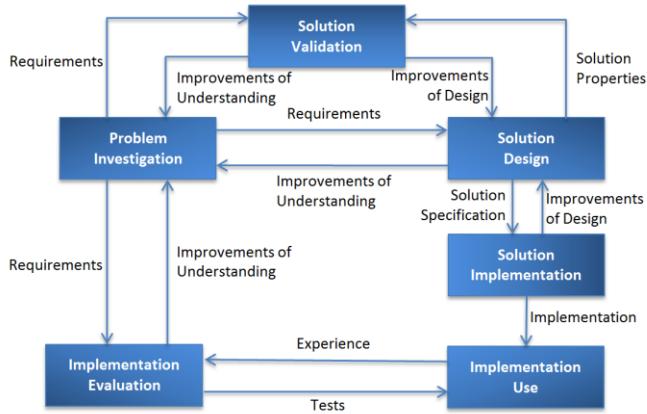


Fig. 7. Activities in the engineering cycle, plus the use activity. Boxes represent activities, arrows represent impacts. There is no preferred sequential relationship among the activities [29]

- 1) **Problem Investigation (PI):** Investigation of the current situation.
 - a) Exploratory Research about Strategic Planning (SP)
 - i) Literature Review
 - ii) Study Cases
 - iii) Analysis of results
 - b) Exploratory Research about Creative Problem Solving (CRE)
 - i) Literature Review
 - ii) Experiments
 - iii) Analysis of results
- 2) **Solution Design (SD):** Propose an improvement to the current situation.
- 3) **Solution Validation (SV):** Investigation of proposed solution properties.
- 4) **Solution Selection (SS):** Selecting the improvements over the recommended ones.
- 5) **Solution Implementation (SI):** Realizing the selected solution, for example, introducing a new RE technique in an organization.
- 6) **Implementation Evaluation (IE):** Investigation of the new situation.
 - a) Comparison with other researches.

XI. CONTRIBUTIONS

In summary, the intended contributions of this Doctoral Research are *Heuristics to prepare Requirements for Evolution based on alternative futures*:

- 1) **Anticipation of possible evolutions:** A study of how *Strategic Management* and *Creativity* can support the identification of alternative futures;
 - a) Expected Result: *Creative Strategic Scenarios (CSS)*;
- 2) **Analysis of risks and impacts of Requirements Changes:** Heuristics to identify Requirements Changes, analyse its risks and impacts and prioritize them;
 - a) Expected Result: *Change Impact Catalogue for i* Models Evolution*: Identification of costs of each kind of changes in evolution of i* models, based on YU (2000).
- 3) **Adaptation of CMsRE to Requirements Changes pre-emptively:** Heuristics to pre-emptively evolve CMsRE, (here represented by scenarios);
 - a) Expected Result: *Change Impact Catalogue for Scenarios Evolution*: Identification of costs of each kind of changes of evolution of scenarios, based on Breitman and Leite.

XII. PROGRESS OF THE RESEARCH

Currently, we are investigating the state of the art and designing a strategy for prediction of organizational changes. This solution aims to prepare the software for requirements changes related to the organizational changes once predicted.

Regarding references of accepted publications we have written, the preliminary aspects of this Doctoral Research were published in the following events:

- **Requirements Engineering at Brazil (ER@BR 2013):** An event co-located with the 21st IEEE International Requirements Engineering Conference (RE'13), it focused on the Brazilian Requirements Engineering Community. All submissions were peer-reviewed and accepted works were published in the CEUR Workshop Proceedings Series. This Doctoral Research was selected as a full paper to be presented in the event. The work was “Requirements Engineering with a Perspective of Software Evolution - Anticipating requirements based on organizational change” [30].
- **First Latin-American School on Software Engineering: Basics and State-of-the-Art (ELAES):** Also co-located with the Latin-American Colloquium on Model-Driven Software Engineering (ESMod), this event happened one week before RE'13, at the same local PUC-Rio. It was four-days of lectures on both basic and state-of-the-art themes of software engineering. The accepted submissions were face-to-face presented to trendsetters in the area and to invited lecturers. This Doctoral Research received insightful feedbacks from senior researchers of Software Engineering Community. The work was “Predicting Requirements Based on Organizational Changes” [31].

ACKNOWLEDGMENT

We would like to thank CAPES for their financial support.

REFERENCES

- [1] A. Etien, C. Rolland, C. Salinesi, "A Meta-modelling Approach to Express Change Requirements", ICSOFT 2006, First International Conference on Software and Data Technologies, 11-14, Setúbal, Portugal, September 2006.
- [2] B. Demil, X. Lecocq, "Business Model Evolution: In Search of Dynamic Consistency", Long Range Planning, 43, n. 2-3, April-June 2010.
- [3] N. A. Ernst, A. Borgida, J. Mylopoulos, "Requirements evolution drives software evolution", Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution, 16-20, Szeged, Hungary, 5-9 September 2011.
- [4] S. Cook, et al. "Evolution in software systems: foundations of the SPE classification scheme", Journal of Software Maintenance and Evolution: Research and Practice, New York, NY, USA, 18, n. 1, 1-35, January 2006.
- [5] M. Lehman, "Program Evolution. Information Processing and Management", Great Britain, 20, n. 1, 19-36, 1984.
- [6] T. Mens et al. "Challenges in Software Evolution", 8th International Workshop on Principles of Software Evolution, Lisbon, Portugal, 5-6 September 2005. pp. 13 - 22.
- [7] N. Maiden, C. Neube, S. Robertson, "Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System", 29th International Conference on Software Engineering, 2007 (ICSE), Minneapolis, MN, 20-26 May 2007. Pp. 632-641.
- [8] N. Maiden, A. Gzikis, S. Robertson, "Provoking Creativity: Imagine What Your Requirements Could Be Like", IEEE Software, v. 21, n. 5, pp. 68-75, September-October 2004.
- [9] N. Maiden, et al. "Requirements Engineering as Creative Problem Solving: A Research Agenda for Idea Finding", 18th IEEE International Requirements Engineering Conference, Sydney, NSW, 27-01 September-October 2010. Pp. 57 - 66.
- [10] D. Bush, A. Finkelstein, "Environmental Scenarios and Requirements Stability", pp. 133-137. 2002.
- [11] J.C.S.P. Leite, A. P. M. Franco, "A strategy for conceptual model acquisition", Requirements Engineering, Proceedings of IEEE International Symposium on IEEE, 1993.
- [12] J.C.S.P. Leite, G. Rossi, F. Balaguer, V. Maiorana, G. Kaplan, G. Hadad, A. Oliveros, "Enhancing a requirements baseline with scenarios", Requirements Engineering 2.4 (1997): pp. 184-198.
- [13] A. G. Sutcliffe, et al. "Supporting scenario-based requirements engineering." Software Engineering, IEEE Transactions on 24.12 (1998): 1072-1088.
- [14] E. Yu, "Modeling Strategic Relationships for Process Reengineering", Ph.D. Dissertation. University of Toronto, Toronto, Ont., Canada, 1996.
- [15] D. Ross, A. Schoman, "Structured analysis for requirements definition", IEEE Transactions on Software Engineering. (Special issue on requirements analysis); vol. 3(1), pp.6-15. 1979.
- [16] N. A. M. Macedo, J. C. S. P. Leite, "Criando uma Arquitetura de Memória Corporativa baseada em um Modelo de Negócio", Pontifical Catholic University of Rio de Janeiro (PUC-Rio). Rio de Janeiro, RJ, Brazil. Doctoral Thesis, p. 172. 2003.
- [17] K. K. Breitman; J. C. S. P. Leite, "A Framework for Scenario Evolution", Proceedings of Third International Conference on Requirements Engineering, Colorado Springs, CO, 214-221, 06-10 April 1998.
- [18] K. K. Breitman, J. C. S. P. Leite, "Evolução De Cenários", Pontifical Catholic University of Rio de Janeiro. Rio de Janeiro - RJ, Brazil. Doctoral Thesis, pp. 162. 2000a.
- [19] K. K. Breitman, J. C. S. P. Leite "Scenario Evolution: a Closer View on Relationships", Proceedings of 4th International Conference on Requirements engineering, Schaumburg, IL, 95-105, 19-23 June 2000b.
- [20] K. K. Breitman; J. C. S. P. Leite "Supporting scenario evolution", Journal of Requirements Engineering, New York, v. 10, n. 2, p. 112 - 131, 2 May 2005.
- [21] J. C. S. P. Leite, et al. "Enhancing a Requirements Baseline with Scenarios", Proceedings of the Third IEEE International Symposium on Requirements Engineering, Annapolis, MD, 6-10 January 1997. pp. 44 - 53.
- [22] N. Maiden, "Framing Requirements Work as Learning. IEEE Software, v. 29, n. 3, p. 8-9, May-June 2012. ISSN 0740-7459.
- [23] C. Rolland, C. Salinesi, A. Etien, "Eliciting gaps in requirements change", Journal Requirements Engineering, New York, Inc. Secaucus, NJ, USA, 9, n. 1, February 2004. 1-15.
- [24] N. A. Ernst, "Software Evolution: a Requirements Engineering Approach", University of Toronto. Toronto, Canada. Doctoral Thesis, p. 255. 2012.
- [25] D. T. Pumareja, "Groupware Requirements Evolution Patterns", University of Twente. Twente, Netherlands, pp. 286. 2013.
- [26] D. Pumareja, K. Sikkel, "An evolutionary approach to groupware implementation: the context of requirements engineering in the socio-technical frame", University of Twente, Centre for Telematics and Information Technology. Enschede, the Netherlands, pp. 1-27. 2002.
- [27] V. Bryl, "Supporting the Design of Socio-Technical Systems by Exploring and Evaluating Design Alternatives", University of Trento. Trento, Italy. Doctoral Thesis, p. 134. 2009a.
- [28] V. Bryl, P. Giorgini, J. Mylopoulos, "Designing socio-technical systems: from stakeholder goals to social networks", Journal Requirements Engineering, London, 14, n. 1, 08 Jan. 2009b. 47-70.
- [29] R. Wieringa, N. Maiden, N. Mead, C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion". Journal Requirements Engineering, New York, Inc. Secaucus, NJ, USA, 11, n. 1, January 2006. 102 - 107.
- [30] M. G. Ferreira, J. C. S. P. Leite, "Requirements Engineering with a Perspective of Software Evolution - Anticipating requirements based on organizational change", Requirements Engineering @ Brazil event (ER@BR 2013), 154-159, Rio de Janeiro, Brazil, 1005, 16 July 2013b.
- [31] M. G. Ferreira, J. C. S. P. Leite, "Predicting Requirements Based on Organizational Changes", Escola Latino Americana de Engenharia de Software (ELA-ES), p. 14, Rio de Janeiro, II, 9-16 July 2013a.

Ready-Set-Transfer! Technology Transfer in the Requirements Engineering Domain (Panel)

Jane Huffman Hayes

Center of Excellence for Software and System Traceability
Computer Science, University of Kentucky
Lexington, USA
hayes@cs.uky.edu

Didar Zowghi

Faculty of Engineering and Information Technology
University of Technology, Sydney
Australia
Didar.Zowghi@uts.edu.au

Abstract—Though the primary goal of requirements engineering research is to propose, develop, and validate effective solutions for important practical problems, practice has shown that successful projects take from 20-25 years to reach full industry adoption, while many projects fade and never advance beyond the initial research phase. In this interactive panel, teams of researchers, representing different requirements engineering research areas, bring ideas for technology transfer to a panel of industrial and government practitioners. The teams make interactive presentations and receive feedback from panelists. Beneath the game-show genre of the panel is the serious goal to foster conversation between practitioners and researchers to improve the effectiveness of technology transfer in the requirements engineering community.

I. INTRODUCTION

Requirements Engineering research topics range from requirements elicitation techniques to studies of formal specification methods in safety critical systems. While these research areas are quite different in nature, they all focus to some extent on improving the ways in which requirements are elicited, analyzed, specified, and managed. Furthermore, they all share the ultimate goal of impacting the state of practice through identifying important research problems, and then proposing, developing, and evaluating novel solutions. Unfortunately, the path from conception of a research idea to full industrial adoption is a challenging one which typically requires the long-term commitment of a community of researchers. To explore these challenges, this panel provides an opportunity for academics and practitioners to engage in an open discussion of the issues affecting the technology adoption path. The panel is designed along the lines of an interactive game-show in which teams of researchers pitch ideas for technology transfer projects to a panel of industrial experts.

II. PHASES OF THE RESEARCH LIFE-CYCLE

A study conducted in the 1980s by Redwine and Riddle followed the maturation path of several software technologies from their conception to their widespread adoption [4]. Redwine et. al found that successful research projects often took from 15-20 years to reach the final phases of adoption. They identified the following six phases of the research process: (i) the basic research stage in which critical research

questions were formulated and foundational ideas and concepts were established; (ii) concept formulation in which a research community was established and started to make headway in addressing a clearly articulated problem, or set of problems; (iii) development and extension, in which there were early signs of adoption and participants worked towards refining and generalizing the approach; (iv) internal enhancement and exploration, in which the approach was applied to other domains and the technology was used in industrial-strength problems; (v) external enhancement and exploration, which extends the previous stage to include a more extensive and broader group of adopters, and further validates the utility and value of the approach; and finally (vi) popularization, in which the tool is fully industrialized for marketing purposes.

Having used these structures to evaluate advances in the software architecture community [5], Mary Shaw discussed several different practices that helped research ideas to mature through these various stages [6]. She noted that research typically proceeds from informal conversations, to position papers, to more formal conference and journal presentations. Furthermore, as the research area matures, it is supported by community activities such as workshops, special topic journal editions, and books. Shaw points out that these activities help to propel the research through the middle phases of the maturation process. As such, most successful research is ultimately dependent upon an active community of researchers who work together over a period of time to solve clearly articulated research problems.

In a panel entitled “What industry wants from research” [1] at the 2011 International Conference on Software Engineering, panelists gave their industrial perspective on the challenges of technology transfer. They highlighted several issues such as the very real financial and career risks that an industrial sponsor incurs when he or she supports a pilot study, and the problem of researchers failing to fully understand the problem or to assess the viability of the solution within a realistic context.

III. MODELS FOR SUCCESSFUL TRANSFER

Wieringa pointed out that research should go far beyond proposing new ideas, which he referred to as design activities, and should provide a rigorous investigation of problems, solutions, and implementations of these solutions, so that

researchers can understand and clearly communicate when and where their proposed solutions can be practically applied [7].

Several researchers have proposed, and implemented, successful technology transfer models along these lines. For example, Gorschek et al. [3] developed a model that assumes close collaboration between industry and academia from the beginning until the end of the research process. Their model involves six steps including area identification, agenda formulation, solution proposal, presentation to practitioners, and full pilot studies. Gorschek et al. have used this model effectively in industry.

There are numerous other models for effective research - industry partnerships. For example, research at the Center of Excellence for Software and Systems Traceability (CoEST) has not only been funded by government agencies such as NASA (US National Aeronautics and Space Administration) and NSF (US National Science Foundation), but has also been sponsored by industries such as Siemens Corporate Research and ABB. This synergy has enabled cutting-edge research ideas to be explored simultaneously with technology transfer in the form of tool development and pilot studies [2]. For example, industrial funding has enabled the development of trace retrieval tools which have been used to conduct industrial pilot studies that correspond to Redwine's development and extension phase of the research lifecycle.

IV. STARTING THE CONVERSATION

If the entire research and publication process is perceived as part of the long-term technology transfer process, then perhaps one of its weaknesses is the lack of conversation between academic researchers and practitioners. This inhibits the research goal of technology transfer as researchers may fail to understand the issues and problems that industry struggles with and as a result may focus their efforts on obscure or unimportant areas of research. On the other hand, once a researcher or group of researchers have identified a non-trivial industrial need, they require a significant incubation period to develop and validate their work before it is ready for industry adoption. While successful projects may take time to come to fruition, long-lived research projects that never make it to market, show no promise of future technology transfer, and produce no useful side benefits are clearly problematic.

Healthy discussions between academics and practitioners are therefore an essential part of the research process, and will help the requirements engineering community to integrate technology transfer plans into the ongoing research process.

V. READY-SET-TRANSFER

In this panel, several different research teams will be invited to present ideas for technology transfer from their own areas of research (through an open call, and then personal invitations if this does not suffice). The teams will likely represent a variety of requirements related research areas such as requirements traceability, creativity in the requirements elicitation process, modeling, formal methods, recommender

systems, and visualization. The panel is set up as a 5-Across competition, often run as part of a Start-Up weekend or other entrepreneurial events. Typically there are 5 teams/entrepreneurs, each given 5 minutes to present their pitches to panel members. Attendees pay a \$5 entrance fee. The winning team (decided by the audience) gets \$500 (hence the 5-Across moniker). In our offering at RE, there will be two or three teams and two rounds. In round one, teams will make 5 minute presentations on their research solution or technology transfer idea (motivating its importance to industry). The panelists will then ask questions/provide feedback for 5 minutes (the panel feedback is designed to present practitioners' perspectives on technology transfer issues). In round two, teams will have 5 minutes devoted to evidence of how the product has been validated and is ready for technology transfer. This will be followed by 5 minutes for panel questions and answers (Q&A), followed by 5 minutes of audience Q&A. Finally, the audience will vote to determine the "winner."

VI. CONCLUSIONS

This panel is designed around the genre of a game show, but it addresses a serious and potentially far-reaching issue affecting the requirements engineering research community. As such it provides an interactive forum for identifying and exploring many critical issues related to technology transfer. Our hope is that the outcome of this panel will strengthen and foster ongoing collaborations between industry and academia that in turn will lead to more effective research projects and successful technology transfers.

REFERENCES

- [1] J. Aranda, D. Damian, M. Petre, M.-A. Storey, and G. Wilson. What industry wants from research. Panel at International Conference on Software Engineering, Hawaii, USA, 2011. J. Clerk Maxwell, *A Treatise on Electricity and Magnetism*, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
- [2] J. Cleland-Huang, A. Czauderna, A. Dekhtyar, O. Gotel, J. Huffman Hayes, E. Keenan, G. Leach, J. Maletic, D. Poshyvanyk, Y. Shin, A. Zisman, G. Antoniol, B. Berenbach, and M. Patrick. Grand challenges, benchmarks, and tracelab: Developing infrastructure for the software traceability research community. Workshop on Traceability in Emerging Forms of Software Eng. (TEFSE), 4(1):17–23, 2011.
- [3] T. Gorschek and C. Wohlin. Packaging software process improvement issues: a method and a case study. *Softw., Pract. Exper.*, 34(14):1311–1344, 2004.
- [4] Samuel T. Redwine Jr. and W. E. Riddle. Software technology maturation. In *ICSE*, pages 189–200, 1985.
- [5] M. Shaw. The coming-of-age of software architecture research. Proc. 23rd International Conference on Software Engineering (ICSE 2001).
- [6] M. Shaw. What makes good research in software engineering? *STTT*, 4(1):1–7, 2002.
- [7] R. Wieringa. Requirements researchers: are we really doing research? *Requir. Eng.*, 10(4):304–306, 2005.

Author Index

- Adedjouma, Morayo 63
Alkhanifer, Abdulrhman 23
Almorsy, Mohamed 315
Alves, Carina 384
Amornborvornwong, Sorawit 123
Amyot, Daniel 73
Anish, Preethu Rose 437
Antón, Annie I. 2, 83
Antonelli, Leandro 263
Aoyama, Mikio 223
Aquanette, Rundale 123
Aydemir, F. Başak 53

Badger, Julia 325
Bano, Muneera 473
Becker, Steffen 344
Bhowmik, Tanmay 133, 243, 467
Böttcher, Boris 329
Bomfim, Camilla 402
Borgida, Alexander 293
Borici, Arber 213
Breaux, Travis D. 163, 193, 273
Briand, Lionel C. 63
Brinkkemper, Sjaak 384
Bruun, Lars 335
Bürger, Jens 103
Buksa, Ilze 489
Bull, Christopher N. 173

Cailliau, Antoine 43
Chakraborty, Rachna 323
Chechik, Marsha 33
Chen, Bihuan 113
Chen, Feng 451
Cheng, Jinghui 13
Chopra, Amit K. 53
Claunch, Charles 325
Cleland-Huang, Jane 123, 253

Dalpiaz, Fabiano 53
Damian, Daniela 213
Daneva, Maya 3
De Gooijer, Thijmen 344
De Oliveira, Tiago 412
Di Sandro, Alessio 33
Duboc, Leticia 402
Dubois, Eric 73

Feja, Sven 333
Ferreira, Marília Guterres 494
Fiedler, Markus 303
Filipovikj, Predrag 444
Fotrousi, Farnaz 303
Fricker, Samuel A. 303
Fuchs, Emmerich 354

Gärtner, Stefan 103
Gervasi, Vincenzo 143
Ghaisas, Smita 364, 437
Ghanavati, Sepideh 73
Ghezzi, Carlo 203

Giorgini, Paolo 53
Glinz, Martin 354
Gordon, David G. 273
Gregoriades, Andreas 317
Grundy, John 315
Guizzardi, Giancarlo 293
Guizzardi, Renata S. S. 293
Guzman, Emitza 153

Hadler, Christian 333
Hammouda, Imed 319
Hansen, Mikkel Bovbjerg 335
Heimann, Virgínia 384
Hess, Anne 354
Hinai, Maryam Al 456
Hogrebe, Frank 329
Horkoff, Jennifer 33, 293
Huffman Hayes, Jane 500

Ingolfo, Silvia 313
Ionita, Dan 485
Iversen, Jørgen Bøndergaard 335

Jansen, Slinger 384
Jørgensen, Jens Bæk 335
Jürjens, Jan 103

Karlsson, Daniel 428
Kauppinen, Marjo 283
King, Jason 183
Knauss, Alessia 213
Knauss, Eric 213, 319
Knudsen, Bjarne 335
Koziolek, Anne 344

Lamsweerde, Axel van 43
Laue, Ralf 329
Lehker, Jean-Michel 193
Leite, Julio Cesar Sampaio do
Prado 263
Letier, Emmanuel 374
Letychevskyi, Oleksandr 331
Li, Feng-Lin 293
Liu, Hui 133
Liu, Lin 293
Liu, Yanji 321
Liwäng, Bo 428
Lönn, Henrik 428
Lohar, Sugandha 123
Lu, Yue 428
Ludi, Stephanie 23
Lundqvist, Kristina 428

Maalej, Walid 153, 364
Männistö, Tomi 283
Mahmoud, Anas 243
Maiden, Neil 394
Martins, Luiz Eduardo Galvão 412
Massacci, Fabio 93
Massey, Aaron K. 83
Menghi, Claudio 203

Mirakhori, Mehdi 253
Mou, Dongyue 327
Mussbacher, Gunter 321
Mylopoulos, John 53, 113, 293, 313

Nguyen, Tuong Huan 315
Niu, Jianwei 193
Niu, Nan 133, 243
Niu, Zhendong 133
Nüttgens, Markus 329
Nunes, Wesley 402
Nuseibeh, Bashar 203
Nyberg, Mattias 444

Oliveros, Alejandro 263
Ott, Greg 123

Pampaka, Maria 317
Pasquale, Liliana 203
Peng, Xin 113
Pinto-Albuquerque, Maria 233
Porter, Chris 374
Pruski, Piotr 123
Putnam, Cynthia 13

Qian, Wenyi 113

Rahimi, Mona 253
Rapp, Daniel 354
Rashid, Awais 233
Rasin, Alexander 123
Ratiu, Daniel 327
Rayson, Paul 173
Riaz, Maria 183
Rifaut, André 73
Rodriguez-Navas, Guillermo 444
Rossi, Gustavo 263
Rubython, Amanda 394
Ruhroth, Thomas 103
Rutledge, Richard L. 83

Sabetzadeh, Mehrdad 63
Saito, Shinobu 223
Salay, Rick 33
Sasse, M. Angela 374
Savolainen, Juha 243
Sawyer, Pete 173
Saxena, Sanjaya Kumar 323
Schaub, Florian 163
Schneider, Kurt 103
Schots, Marcelo 402
Seyff, Norbert 354
Siena, Alberto 313
Singh, Munindar P. 53
Slankas, John 183
Slavin, Rocky 193
Speck, Andreas 333
Spörri, Peter 354
Stålhane, Tor 420
Su, Yukun 321
Sutcliffe, Alistair 173, 317
Swire, Peter P. 83

Takeuchi, Mutsuki 223
Teruel, Miguel A. 461
Throop, David 325

| | | | | | |
|---------------------------|-----|----------------------------|-----|----------------------|----------|
| Töhönen, Harri | 283 | Wasserman, Anthony I. | 1 | Yamada, Setsuo | 223 |
| Tran, Le Minh Sang | 93 | Weigert, Thomas | 331 | Yin, Xinshang | 321 |
| Tsigkanos, Christos | 203 | Wien, Tormod | 420 | Zhao, Wenyun | 113 |
| Valen a, George | 384 | Williams, Laurie | 183 | Zhou, Jiale | 428, 479 |
| Wang, Huanhuan | 113 | Witt, S ren | 333 | Zhu, Xiuna | 327 |
| | | Wohlrab, Rebekka | 344 | Zowghi, Didar | 143, 500 |