

Selmin Nurcan Henderik A. Proper
Pnina Soffer John Krogstie
Rainer Schmidt Terry Halpin
Ilia Bider (Eds.)

Enterprise, Business-Process and Information Systems Modeling

14th International Conference, BPMDS 2013

18th International Conference, EMMSAD 2013

Held at CAiSE 2013, Valencia, Spain, June 2013, Proceedings



Springer

Lecture Notes in Business Information Processing

147

Series Editors

Wil van der Aalst

Eindhoven Technical University, The Netherlands

John Mylopoulos

University of Trento, Italy

Michael Rosemann

Queensland University of Technology, Brisbane, Qld, Australia

Michael J. Shaw

University of Illinois, Urbana-Champaign, IL, USA

Clemens Szyperski

Microsoft Research, Redmond, WA, USA

Selmin Nurcan Henderik A. Proper
Pnina Soffer John Krogstie
Rainer Schmidt Terry Halpin
Ilia Bider (Eds.)

Enterprise, Business-Process and Information Systems Modeling

14th International Conference, BPMDS 2013
18th International Conference, EMMSAD 2013
Held at CAiSE 2013, Valencia, Spain, June 17-18, 2013
Proceedings

Volume Editors

Selmin Nurcan

University Paris 1 Pantheon-Sorbonne, Paris, France

E-mail: nurcan@univ-paris1.fr

Henderik A. Proper

Public Research Centre - Henri Tudor, Luxembourg-Kirchberg, Luxembourg

E-mail: e.proper@tudor.lu

Pnina Soffer

University of Haifa, Israel

E-mail: spnina@is.haifa.ac.il

John Krogstie

Norwegian University of Science and Technology, Trondheim, Norway

E-mail: krogstie@idi.ntnu.no

Rainer Schmidt

Aalen University, Germany

E-mail: rainer.schmidt@htw-aalen.de

Terry Halpin

INTI International University, Kuala Lumpur, Malaysia

E-mail: terry.halpin@logicblox.com

Ilia Bider

Stockholm University/IbisSoft, Sweden

E-mail: ilia@ibisoft.se

ISSN 1865-1348

e-ISSN 1865-1356

ISBN 978-3-642-38483-7

e-ISBN 978-3-642-38484-4

DOI 10.1007/978-3-642-38484-4

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013938407

ACM Computing Classification (1998): J.1, H.4.1, H.3.5, D.2

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This book contains the proceedings of two long-running events held along with the CAiSE conferences relating to the areas of enterprise, business-process and information systems modeling: the 14th International Conference on Business Process Modeling, Development and Support (BPMDS 2013), and the 18th International Conference on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2013). The two working conferences are introduced below.

BPMDS 2013

BPMDS has been held as a series of workshops devoted to business process modeling, development, and support since 1998. During this period, business process analysis and design has been recognized as a central issue in the area of information systems (IS) engineering. The continued interest in these topics on behalf of the IS community is reflected by the success of the last BPMDS events and the recent emergence of new conferences and workshops devoted to the theme. In 2011, BPMDS became a two-day working conference attached to CAiSE (Conference on Advanced Information Systems Engineering). The basic principles of the BPMDS series are:

1. BPMDS serves as a meeting place for researchers and practitioners in the areas of business development and business applications (software) development.
2. The aim of the event is mainly discussions, rather than presentations.
3. Each event has a theme that is mandatory for idea papers.
4. Each event's results are, usually, published in a special issue of an international journal.

The goals, format, and history of BPMDS can be found on the website: <http://www.bpmds.org/>

The intention of BPMDS is to solicit papers related to business process modeling, development, and support (BPMDS) in general, using quality as a main selection criterion. As a working conference, we aim to attract papers describing mature research, but we still give place to industrial reports and visionary idea papers. To encourage new and emerging challenges and research directions in the area of business process modeling, development, and support, we have a unique focus theme every year. Papers submitted as idea papers are required to be of relevance to the focus theme, thus providing a mass of new ideas around a relatively narrow but emerging research area. Full research papers and experience reports do not necessarily need to be directly connected to this theme (but they should still be explicitly relevant to BPMDS). The focus theme for BPMDS 2013 idea papers was "Coping with Complexity in Business Processes." Today,

business processes have to cope with increasing complexity in several areas. First, business processes are increasingly becoming cross and inter-organizational. External processes and services have to be integrated into processes. Second, data are becoming more and more important for business processes. Recent advances such as data science and big data provide huge amounts of information to be processed in business processes. Third, business processes are executed in complex environments, which may consist of cloud services and resources and may introduce flexibility requirements. Other considerations such as data orientation as in MDM (master data management) in relation to big data, need for context-awareness and flexibility, integration of business processes with social media also set new challenges for coping with complexity in business processes.

BPMDS 2013 received 54 submissions from 24 countries (Argentina, Austria, Belgium, Brazil, Canada, Colombia, France, Germany, Greece, Israel, Italy, Latvia, Mexico, The Netherlands, Norway, Portugal, Spain, Sweden, Switzerland, Tunisia, Turkey, Uganda, UK, USA). The management of paper submission and reviews was supported by the EasyChair conference system. Selecting the papers to be accepted was a worthwhile effort. Each paper received at least three reviews. Eventually, 20 high-quality papers were selected; among them two experience reports and three idea papers. The accepted papers cover a wide spectrum of issues related to business process development, modeling, and support. They are organized under the following section headings:

- Innovative representations for knowledge-intensive processes
- Business process management in practice
- Analysis of business process models
- Model-based business process analysis
- Flexible business process management
- Improvement and change patterns
- Process model repositories

We wish to thank all the people who submitted papers to BPMDS 2013 for having shared their work with us, as well as the members of the BPMDS 2013 Program Committee, who made a remarkable effort reviewing the large number of submissions. We also thank the organizers of CAiSE 2013 for their help with the organization of the event, and IFIP WG8.1 for the support.

April 2013

Selmin Nurcan
Pnina Soffer
Rainer Schmidt
Ilia Bider

Organization

Program Committee

Sebastian Adam	Fraunhofer IESE, Germany
Antonia Albani	University of St. Gallen, Switzerland
Joao Paulo Almeida	Federal University of Espirito Santo, Brazil
Eric Andonoff	IRIT/University Toulouse 1, France
Renata Araujo	UNIRIO, Brazil
Judith Barrios Albornoz	University of Los Andes
Ilia Bider	Stockholm University/IbisSoft
Karsten Boehm	FH KufsteinTirol - University of Applies Science, Austria
Pere Botella	Universitat Politecnica de Catalunya, Spain
Dirk Fahland	Technische Universiteit Eindhoven, The Netherlands
Luciano García-Bañuelos	University of Tartu, Estonia
Johny Ghattas	University of Haifa, Israel
Claude Godart	Loria, France
Giancarlo Guizzardi	Federal University of Espirito Santo (UFES), Brazil
Marta Indulska	The University of Queensland, Australia
Paul Johannesson	Royal Institute of Technology
Marite Kirikova	Riga Technical University, Latvia
Agnes Koschmider	Karlsruhe Institute of Technology, Germany
Marcello La Rosa	Queensland University of Technology, Australia
Jan Mendling	Wirtschaftsuniversität Wien, Austria
Bela Mutschler	University of Applied Sciences Ravensburg-Weingarten, Germany
Jens Nimis	University of Applied Sciences Karlsruhe, Germany
Selmin Nurcan	Université de Paris 1 Panthéon - Sorbonne, France
Andreas Oberweis	Universität Karlsruhe, Germany
Oscar Pastor Lopez	Universitat Politecnica de Valencia, Valencia
Gil Regev	Ecole Polytechnique Fédérale de Lausanne, Switzerland
Manfred Reichert	University of Ulm, Germany

VIII Organization

Hajo A. Reijers	Eindhoven University of Technology, The Netherlands
Iris Reinhartz-Berger	University of Haifa, Israel
Stefanie Rinderle-Ma	University of Vienna, Austria
Colette Rolland	Université de Paris 1 Panthéon - Sorbonne, France
Michael Rosemann	Queensland University of Technology, Australia
Shazia Sadiq	The University of Queensland, Australia
Rainer Schmidt	Aalen University, Germany
Pnina Soffer	University of Haifa, Israel
Lars Taxén	Linköping University, Sweden
Roland Ukor	FirstLinq Ltd.
Barbara Weber	Univ. of Innsbruck, Austria
Matthias Weidlich	Technion - Israel Institute of Technology, Israel
Jelena Zdravkovic	Stockholm University, Sweden

Additional Reviewers

Conforti, Raffaele
Domigall, Yannic
Dunkl, Reinhold
Fdhila, Walid
Grambow, Gregor
Hess, Anne
Hildebrandt, Tobias
Knuplesch, David
Leitner, Maria
Lohrmann, Matthias
Perjons, Erik
Polyvyanyy, Artem
Raber, David
Schickler, Marc
Schobel, Johannes
Ünalan, Özgür

Preface

EMMSAD 2013

The field of information systems analysis and design includes numerous information modelling methods and notations (e.g., ER, ORM, UML, Archimate, EPC, DEMO, DFDs, BPMN) that are typically evolving. Even with some attempts toward standardization (e.g., UML for object-oriented design), new modelling methods are constantly being introduced, many of which differ only marginally from existing approaches. These ongoing changes significantly impact the way information systems are being analyzed and designed in practice. EMMSAD focuses on exploring, evaluating, and enhancing current information modelling methods and methodologies. Though the need for such studies is well recognized, there is a paucity of such research in the literature. The objective of the conference is to provide a forum for researchers and practitioners interested in modelling methods in systems analysis and design to meet and exchange research ideas and results. It also gives the participants an opportunity to present their research papers and experience reports, and to take part in open discussions. EMMSAD 2013 was the 18th in a series of events, previously held in Heraklion, Barcelona, Pisa, Heidelberg, Stockholm, Interlaken, Toronto, Velden, Riga, Porto, Luxembourg, Trondheim, Montpellier, Amsterdam, Hammamet, London, and Gdansk. This year we had 27 submissions with authors from 18 countries and six continents (Australia, Austria, Belgium, Canada, France, Germany, Israel, Latvia, Luxembourg, Malaysia, Morocco, The Netherlands, Norway, Portugal, Spain, Sweden, Tunisia, and USA). After an extensive review process by a distinguished international Program Committee, with each paper receiving at least three reviews, we accepted the ten full papers and two short papers that appear in these proceedings. Congratulations to the successful authors! Apart from the contribution by paper authors, the quality of EMMSAD 2013 depended in no small way on the generous contribution of time and effort by the Program Committee and the additional reviewers. Their work is greatly appreciated. We also express our sincere thanks to the CAiSE Organizing Committee. Continuing with our very successful collaboration with IFIP WG 8.1 (research.idi.ntnu.no/ifip-wg81) that started in 1997, this year's event was again a joint activity between CAiSE and WG 8.1. Other co-organizers this year were the Enterprise Architecture Network, The ORM Foundation, and the Enterprise Engineering Team.

For more information on the EMMSAD-series, see our website:
www.emmsad.org

April 2013

Henderik A. Proper
John Krogstie
Terry Halpin

Organization

EMMSAD Organizing Committee

Henderik A. Proper	CRP Henri Tudor, Luxembourg, and Radboud University Nijmegen, The Netherlands
Terry Halpin	INTI International University, Malaysia and LogicBlox, Australia
John Krogstie	Norwegian Institute of Science and Technology, Norway
Keng Siau	Missouri University of Science and Technology, USA

EMMSAD 2013 Program Committee

Stephan Aier	University of St. Gallen, Switzerland
Antonia Albani	Delft University of Technology, The Netherlands
Raijan Ali	Lero, University of Limerick, Ireland
Eduard Babkin	Higher School of Economics, Moscow, Russia
Herman Balsters	University of Groningen, The Netherlands
Annie Becker	Florida Institute of Technology, USA
Giuseppe Berio	University of Turin, Italy
Nacer Boudjilida	Loria, France
Andy Carver	INTI International University, Malaysia
Olga De Troyer	Vrije Universiteit Brussel, Belgium
John Erickson	University of Nebraska-Omaha, USA
Peter Fettke	Institute for Information Systems (IWi) at the DFKI, Germany
Remigijus Gustas	Karlstad University, Sweden
Frank Harmsen	Maastricht University and E&Y Advisory, The Netherlands
Wolfgang Hesse	University of Marburg, Germany
Stijn Hoppenbrouwers	HAN University of Applied Sciences, Arnhem, The Netherlands
Jon Iden	Norges Handelshøyskole, Bergen, Norway
Marite Kirikova	Riga Technical University, Latvia
Bogdan Lent	University of Applied Sciences, Zurich, Switzerland
Pericles Loucopoulos	Loughborough University, UK
Kalle Lyytinen	Case Western Reserve University, USA

Leszek Maciaszek	Wroclaw University of Economics, Poland and Macquarie University Sydney, Australia
Florian Matthes	Technical University of Munich, Germany
Raimundas Matulevičius	University of Tartu, Estonia
Graham McLeod	University of Cape Town, South Africa
Jan Mendling	Humboldt University, Berlin
Wolfgang Molnar	CRP Henri Tudor, Luxembourg
Tony Morgan	INTI International University, Malaysia
Haralambos Mouratidis	University of East London, UK
Andreas L. Opdahl	University of Bergen, Norway
Sietse Overbeek	University of Duisburg-Essen, Germany
Hervé Panetto	University Henri Poincaré Nancy I, France
Barbara Pernici	Politecnico di Milano, Italy
Anne Persson	University of Skövde, Sweden
Michaël Petit	University of Namur, Belgium
Georgios Plataniotis	CRP Henri Tudor, Luxembourg
Nava Pliskin	Ben-Gurion University of the Negev, Israel
Paul Ralph	Lancaster University, UK
Jolita Ralyté	University of Geneva, Switzerland
Sudha Ram	University of Arizona, USA
Jan Recker	Queensland University of Technology, Australia
Colette Rolland	University of Paris 1, France
Michael Rosemann	Queensland University of Technology, Australia
Matti Rossi	Helsinki School of Economics, Finland
Kurt Sandkuhl	University of Rostock, Germany
Peretz Shoval	Ben-Gurion University of the Negev, Israel
Piotr Soja	Cracow Economic University, Poland
Janis Stirna	Royal Institute of Technology, Sweden
Inge van de Weerd	Utrecht University, The Netherlands
Wil van der Aalst	Eindhoven University, The Netherlands
Dirk van der Linden	CRP Henri Tudor, Luxembourg
Johan Versendaal	University of Utrecht, The Netherlands
Carson Woo	University of British Columbia, Canada
Jelena Zdravkovic	Stockholm University, Stockholm, Sweden
Martin Zelm	CIMOSA, Germany
Iryna Zolotaryova	Kharkiv National University of Economics, Ukraine
Pär Ågerfalk	Uppsala University, Sweden

EMMSAD 2013 Additional Reviewers

Stefan Bischoff	University of St. Gallen, Switzerland
Sabine Buckl	Technical University of Munich, Germany
Sybren de Kinderen	CRP Henri Tudor, Luxembourg

Mario Lezoche	Université Henri Poincaré - Nancy 1, Nancy, France
Eduardo Loures	Pontifícia Universidade Católica do Paraná, Brazil
Alexander W. Schneider Jürgen Walter	Technical University of Munich, Germany
Simon Weiss	DFKI, Germany
Xinwei Zhu	University of St. Gallen, Switzerland
Jan Pieter Zwart	Queensland University of Technology, Brisbane, Australia
	HAN University of Applied Sciences, Arnhem, The Netherlands

Table of Contents

BPMDS and EMMSAD Session 1: Joint Keynote

Multi-level Modelling: Time for a 'Copernican' (R)Evolution?: (Keynote)	1
<i>Brian Henderson-Sellers</i>	

BPMDS Session 2: Innovative Representations for Knowledge-Intensive Processes

Making Sense of Declarative Process Models: Common Strategies and Typical Pitfalls	2
<i>Cornelia Haisjackl, Stefan Zugal, Pnina Soffer, Irit Hadar, Manfred Reichert, Jakob Pinggera, and Barbara Weber</i>	

Blending BPMS with Social Software for Knowledge-Intense Work: Research Issues	18
<i>Nancy Alexopoulou, Mara Nikolaïdou, and Christian Stary</i>	

Context-Aware Agile Business Process Engine: Foundations and Architecture	32
<i>Irina Rychkova, Manuele Kirsch-Pinheiro, and Bénédicte Le Grand</i>	

BPMDS Session 3: BPM in Practice

Business Process Workarounds: What Can and Cannot Be Detected by Process Mining	48
<i>Nesi Outmazgin and Pnina Soffer</i>	

Using Data-Centric Business Process Modeling for Discovering Requirements for Business Process Support Systems: Experience Report	63
<i>Ilia Bider, Erik Perjons, and Zakria Riaz Dar</i>	

A Methodological Framework with Lessons Learned for Introducing Business Process Management	78
<i>Sebastian Adam, Norman Riegel, and Matthias Koch</i>	

BPMDS Session 4: Analysis of Business Process Models

Repairing Business Process Models as Retrieved from Source Code	94
<i>María Fernández-Ropero, Hajo A. Reijers, Ricardo Pérez-Castillo, and Mario Piattini</i>	

Bridging Abstraction Layers in Process Mining: Event to Activity Mapping	109
--------------------------------------------------------------------------------	-----

Thomas Baier and Jan Mendling

Improving Business Process Models with Agent-Based Simulation and Process Mining	124
----------------------------------------------------------------------------------------	-----

Fernando Szymanski, Célia G. Ralha, Gerd Wagner, and Diogo R. Ferreira

BPMDS Session 5: Model-Based Business Process Analysis

Towards a Framework for Modeling Business Compensation Processes	139
------------------------------------------------------------------------	-----

Anis Boubaker, Hafedh Mili, Yasmine Charif, and Abderrahmane Leshob

An Effort Prediction Model Based on BPM Measures for Process Automation	154
-------------------------------------------------------------------------------	-----

Banu Aysolmaz, Deniz İren, and Onur Demirörs

On the Use of Goal Models and Business Process Models for Elicitation of System Requirements	168
----------------------------------------------------------------------------------------------------	-----

Jose Luis de la Vara, Juan Sánchez, and Oscar Pastor

BPMDS Session 6: Flexible BPM

Multi-level Autonomic Business Process Management	184
---------------------------------------------------------	-----

Karolyne Oliveira, Jaelson Castro, Sergio España, and Oscar Pastor

Multi-perspective Business Process Monitoring	199
-----------------------------------------------------	-----

Amin Jalali and Paul Johannesson

Corrective Evolution of Adaptable Process Models	214
--------------------------------------------------------	-----

Luciano Baresi, Annappaola Marconi, Marco Pistore, and Adina Sirbu

BPMDS Session 7: Improvement and Change Patterns

Demonstrating the Effectiveness of Process Improvement Patterns	230
-----------------------------------------------------------------------	-----

Matthias Lohrmann and Manfred Reichert

Enhancing Modeling and Change Support for Process Families through Change Patterns	246
------------------------------------------------------------------------------------------	-----

Clara Ayora, Victoria Torres, Barbara Weber, Manfred Reichert, and Vicente Pelechano

Change Patterns in Use: A Critical Evaluation	261
<i>Barbara Weber, Jakob Pinggera, Victoria Torres, and Manfred Reichert</i>	

BPMDS Session 8: Process Model Repositories

Synthesizing a Library of Process Templates through Partial-Order Planning Algorithms	277
<i>Andrea Marrella and Yves Lespérance</i>	

Spotting Terminology Deficiencies in Process Model Repositories	292
<i>Fabian Pittke, Henrik Leopold, and Jan Mendling</i>	

EMMSAD Session 2: Advanced Modelling

Modeling of Reference Schemes	308
<i>Terry Halpin</i>	

Visually Capturing Usage Context in BPMN by Small Adaptations of Diagram Notation	324
<i>Guttorm Sindre, John Krogstie, and Sundar Gopalakrishnan</i>	

EMMSAD Session 3: Capturing Design Knowledge

Capturing Decision Making Strategies in Enterprise Architecture – A Viewpoint	339
<i>Georgios Plataniotis, Sybren de Kinderen, and Henderik A. Proper</i>	

Constructing Domain Knowledge through Cross Product Line Analysis	354
<i>Ora Wulf-Hadash and Iris Reinhartz-Berger</i>	

EMMSAD Session 4: Method Engineering

Incremental Method Enactment for Computer Aided Software Engineering Tools	370
<i>Kevin Vlaanderen, Geurt van Tuyl, Sjaak Brinkkemper, and Slinger Jansen</i>	

Gamification to Support the Run Time Planning Process in Adaptive Case Management	385
<i>Danny Oldenhove, Stijn Hoppenbrouwers, Theo van der Weide, and Remco Lagarde</i>	

EMMSAD Session 5: Modelling Process

- A Semiotic Approach to Data Quality 395
John Krogstie

- Feedback-Enabled MDA-Prototyping Effects on Modeling Knowledge 411
Gayane Sedrakyan and Monique Snoeck

EMMSAD Session 6: Specialized Modelling

- A Rigorous Reasoning about Model Transformations Using the B Method 426
Akram Idani, Yves Ledru, and Adil Anwar

- Managing Run-Time Variability in Robotics Software by Modeling Functional and Non-functional Behavior 441
Alex Lotz, Juan F. Inglés-Romero, Cristina Vicente-Chicote, and Christian Schlegel

EMMSAD Session 7: Modelling Experiences

- The World Out There: From Systems Modelling to Enterprise Modelling 456
Sobah Abbas Petersen and John Krogstie

- Process Mining Versus Intention Mining 466
Ghazaleh Khodabandeh, Charlotte Hug, Rebecca Deneckère, and Camille Salinesi

- Author Index** 481

Multi-level Modelling: Time for a ‘Copernican’ (R)Evolution? (Keynote)

Brian Henderson-Sellers

University of Technology, Sydney
Australia

Contemporary modelling practice links models of business system concepts to metamodels to metametamodels in a multilayer architecture. One standard architecture, that of the Object Management Group, utilizes so-called strict metamodeling, which can readily be shown to lead to several concerns, paradoxes and challenges within modelling approaches, including MDE, currently in use. Utilizing an analogy in the paradigm shift from a Ptolemaic model of the solar system to a Copernican understanding, I will argue that the current state of affairs in conceptual modelling, as used in developing information systems and modelling business processes, is ‘Ptolemaic’ and, using ideas from language use, ontology engineering and philosophy, suggest a framework for ‘Copernican’ modelling that will provide both a paradigm shift in multilevel modelling and also a new orthodoxy for the future that will ensure simpler and more satisfying modelling solutions for both business processes and software development in the years to come.

Making Sense of Declarative Process Models: Common Strategies and Typical Pitfalls^{*}

Cornelia Haisjackl¹, Stefan Zugal¹, Pnina Soffer², Irit Hadar²,
Manfred Reichert³, Jakob Pinggera¹, and Barbara Weber¹

¹ University of Innsbruck, Austria

{cornelia.haisjackl,stefan.zugal,jakob.pinggera,barbara.weber}@uibk.ac.at

² University of Haifa, Israel

{spnina,hadar}@is.haifa.ac.il

³ University of Ulm, Germany

manfred.reichert@uni-ulm.de

Abstract. Declarative approaches to process modeling are regarded as well suited for highly volatile environments as they provide a high degree of flexibility. However, problems in understanding and maintaining declarative business process models impede often their usage. In particular, how declarative models are understood has not been investigated yet. This paper takes a first step toward addressing this question and reports on an exploratory study investigating how analysts make sense of declarative process models. We have handed out real-world declarative process models to subjects and asked them to describe the illustrated process. Our qualitative analysis shows that subjects tried to describe the processes in a *sequential way* although the models represent circumstantial information, namely, conditions that produce an outcome, rather than a sequence of activities. Finally, we observed difficulties with single building blocks and combinations of relations between activities.

Keywords: Declarative Process Models, Empirical Research, Understandability.

1 Introduction

Regarding the analysis and design of information systems, conceptual modeling has proven to foster understanding and communication [1]. For example, *business process models* (*process models* for short) have been employed in the context of process-aware information systems, service-oriented architectures, and web services [2]. Recently, *declarative* process models have gained attention due to their flexibility with respect to modeling and execution of processes [3]. While technical issues of declarative process modeling, such as formalization of semantics [4], maintainability [5], verification [6], and execution [7] are well understood, understandability issues of declarative models have not been investigated in detail yet.

* This research is supported by Austrian Science Fund (FWF): P23699-N23 and the BIT fellowship program.

In particular, it has been argued that understandability may be hampered by lack of computational offloading [8] or hidden dependencies [9]. Put differently, it is not entirely clear whether the full potential of declarative modeling can be exploited or whether understandability issues will interfere.

We approach these issues by studying the sense-making of declarative process models through the lens of an empirical investigation. In particular, we handed out declarative process models to subjects and asked them to describe the illustrated process. In addition, we asked them to voice their thoughts while describing the process, i.e., we applied *think-aloud techniques* [10] to get insights into the subject's reasoning processes. Since we were interested in how different structures of the process representation would influence process model understandability, we maintained another variant of each model describing the same process, but making use of modularization, i.e., sub-processes. The contribution of this work is twofold. On one hand, we provide insights into how subjects make sense of declarative process models, e.g., we analyze strategies how to read declarative process models. On the other, we consider characteristic problems that occur when scanning declarative process models. Our contribution aims at guiding the future development of supporting tools for system analysts, as well as pointing out typical pitfalls to teachers and educators of analysts.

The exploratory study reported in this paper is part of a larger investigation on declarative process models. While our previous work focused on quantitative results, this paper deals with qualitative data solely.¹ Sect. 2 gives background information. Sect. 3 describes the setup of the exploratory study, whereas Sect. 4 deals with its execution. Sect. 5 presents the results of the exploratory study and Sect. 6 a corresponding discussion. Related work is presented in Sect. 7. Finally, Sect. 8 concludes the paper.

2 Background: Declarative Process Models

There has been a long tradition of modeling business processes in an imperative way. Process modeling languages supporting this paradigm, like BPMN and EPC, are widely used. Recently, *declarative approaches* have received increasing interest, as they suggest a fundamentally different way of describing business processes [6]. While imperative models specify exactly *how* things must be done, declarative approaches focus on the logic that governs the interplay of process actions by describing *activities* that may be performed, as well as *constraints* prohibiting undesired behavior. Constraints found in literature may be divided into existence constraints, relation constraints, and negation constraints [11]. *Existence constraints* specify how often an activity must be executed for one particular process instance. In turn, *relation constraints* restrict the ordering of activities by imposing respective restrictions. Finally, *negation constraints* define negative relations between activities. Table 1 shows examples for each category, an overview of all constraints can be found in [11].

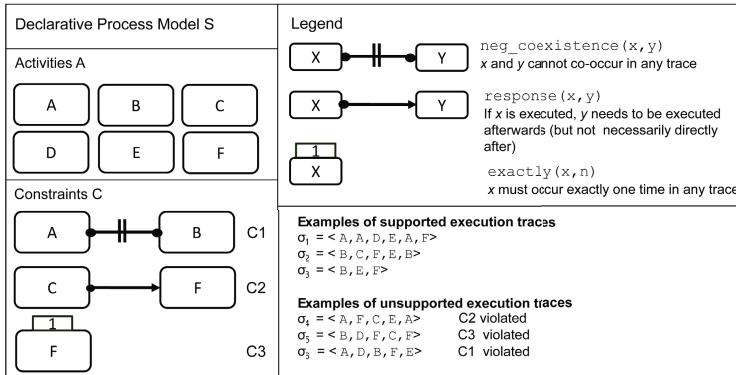
¹ The exploratory study's material can be downloaded from:

<http://bpm.q-e.at/experiment/HierarchyDeclarative>

Table 1. Definition of constraints

Group	Constraint	Definition
existence	exactly(a,n)	activity a must occur exactly n times
	existence(a,n)	a must occur at least n times
	init(a)	a must be the first executed activity in every trace
	last(a)	a must be the last executed activity in every trace
relation	precedence(a,b)	activity b must be preceded by activity a (but not necessarily directly preceded)
	response(a,b)	if a is executed, b must be executed afterwards (but not necessarily directly afterwards)
	chain_response(a,b)	if a is executed, b is executed directly afterwards
	coexistence(a,b)	if a is executed, b must be executed and vice-versa
negation	neg_response(a,b)	if a is executed, b must not be executed afterwards
	neg_coexistence(a,b)	a and b cannot co-occur in any trace

An example of a declarative process model S specified with ConDec [6] is depicted in Fig. 1. The model consists of six distinct activities A, B, C, D, E, and F. In addition, it comprises three constraints. The *neg_coexistence* constraint, i.e., C1, forbids that A and B co-occur in the same trace. In turn, the *response* constraint, i.e., C2, requires that every execution of C must be followed by one of F before the process instance may complete. Finally, the *exactly* constraint, i.e., C3, states that F must be executed exactly once per process instance. While instances with traces $\sigma_1 = \langle A, A, D, E, A, F \rangle$, $\sigma_2 = \langle B, C, F, E, B \rangle$, and $\sigma_3 = \langle B, E, F \rangle$ satisfy all the constraints, $\sigma_4 = \langle A, F, C, E, A \rangle$ violates C2, $\sigma_5 = \langle B, D, F, C, F \rangle$ violates C3, and $\sigma_6 = \langle A, D, B, F, E \rangle$ violates C1. $\sigma_5 = \langle B, D, F, C, F \rangle$ highlights a *hidden dependency* between C and F. The combination of the *exactly* constraint, i.e., C3, and the *response* constraint, i.e., C2, adds an implicit constraint that does not exist when looking at the constraints in isolation. This hidden dependency prohibits that F is executed before C, assuming that C is executed at all.

**Fig. 1.** Example of a declarative process model

Hierarchy in Declarative Process Models. Using modularization to hierarchically structure information has been identified as a viable approach to deal with complexity for decades [12]. Taking a look at declarative process models with hierarchy in general, a sub-process may be introduced in a process model via a *complex activity*, referring to a process model. When the complex activity is executed, the referred process model, i.e., the sub-process, is instantiated (see [13] for details). Fig. 2a) shows a hierarchical model, complex activity *B* refers to a sub-process that contains activities *C* and *D*. Fig. 2b) shows the corresponding flat process model. Even though Fig. 2a) and Fig. 2b) are semantically equivalent, they differ in the number of activities and constraints.

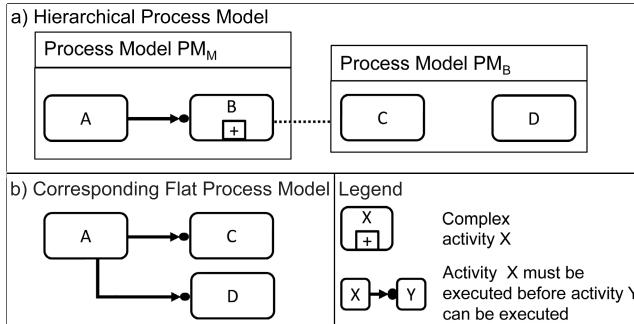


Fig. 2. Example of a process model with and without hierarchy

3 Defining and Planning the Exploratory Study

In order to investigate how subjects make sense of declarative process models we conduct an exploratory study. In particular, we are interested in common strategies and typical pitfalls occurring during this sense-making process. Since there has been no considerable research on understandability issues of declarative process models, and hence no theories exist we can base our investigation on, we address the topic in an exploratory manner using a qualitative research approach [14]. In particular, we use the think-aloud method, i.e., we ask participating subjects to voice their thoughts, allowing for a detailed analysis of their reasoning process [10]. Then, we turn to grounded theory [15], an analysis approach for identifying recurring aspects and grouping them to categories. These categories are validated and refined throughout the analysis process. First of all, we describe setup and planning of the exploratory study.

Subjects. In order to ensure that obtained results are not influenced by unfamiliarity with declarative process modeling, subjects need to be sufficiently trained. Even though we do not require experts, subjects should have at least a moderate understanding of declarative processes' principles.

Objects. The process models used in the study originate from a case study [16] and describe real-world business processes. From a set of 24 process models collected in this case study, 4 models were chosen as basic *objects* for the exploratory study. This was accomplished in a way ensuring that the numbers of activities and constraints vary. To make the models amenable for this study, they underwent the following procedure. First, the models were translated to English (the case study was conducted in German) since all exercises were done in English (four subjects did not speak German). Second, since the models collected during the modeling sessions had not gone through quality assessment, they were scanned for errors and corrected accordingly. Third, since we were interested in how different structures of the process representation would influence the process models' understandability, we created a second variant of each process describing the same process, but making use of sub-processes. Consequently, we have two variants of each process model: a flat and a hierarchical one.

Table 2. Characteristics of the process models used in this study

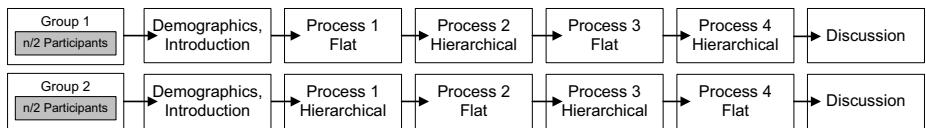
	Type	Proc. 1	Proc. 2	Proc. 3	Proc. 4
Activities	flat	11	8	23	23
	hierarchy	13	9	26	26
Constraints	flat	19	7	30	45
	hierarchy	21	9	28	44
Constr. types		8	4	7	5
Sub-processes	hierarchy	2	1	3	2
Components		2	5	2	2
Domain		Software development	Teaching	Electronic company	Buying an apartment

Table 2 summarizes the characteristics of the process models. The latter comprise between 8 and 26 activities, and between 7 and 45 constraints. The differences in the number of activities between flat and hierarchical models are caused by the complex activities representing sub-processes in the hierarchical models (cf. Sect. 2). Similarly, constraints had to be added or removed to preserve the behavior when creating a hierarchical model. Process models vary regarding the degree of interconnectivity of constraints, i.e., models consist of two to five components (cf. Table 2). A component is defined as a part of the model where any two activities are connected by constraints, and not connected to any other activity in the model. The process models are based on four different domains describing bug fixing in a software company, a teacher's preparations prior to teaching, a worker's duties at an electronic company, and buying and renovating an apartment (cf. Table 2). The process models contain constraints of all three types, i.e., existence, relation, and negation constraints, except the second process model (no negation constraints). Table 3 provides additional information on the constraint types included in each process model.

Table 3. Constraints of the process models used in this study

Group	Constraint	flat				hierarchical			
		P 1	P 2	P 3	P 4	P 1	P 2	P 3	P 4
existence	existence constraints	5	2	1	10	7	4	1	13
	init	1	1	1	0	1	1	1	0
	last	0	1	0	1	0	1	0	1
relation	precedence	4	3	18	20	4	3	18	20
	response	0	0	1	0	0	0	1	0
	succession	0	0	0	4	0	0	0	4
	coexistence	0	0	1	0	0	0	1	0
	chained precedence	2	0	0	0	2	0	0	0
	chained response	3	0	1	0	3	0	1	0
	chained succession	1	0	0	0	1	0	0	0
negation	negation response	2	0	0	10	2	0	0	6
	mutual exclusion	1	0	7	0	1	0	5	0

Design. Fig. 3 shows the overall design of the exploratory study: First, subjects are *randomly* assigned to two groups of similar size. Regardless of the group assignment, demographical data is collected and subjects obtain introductory assignments. To support subjects in their task, cheat sheets briefly summarizing the constraints’ semantics, are provided, which can be used throughout the study. Introductory tasks allow subjects to familiarize themselves with the type of tasks to be performed—potential problems can therefore be resolved at this stage without influencing actual data collection. After this familiarization phase, subjects are confronted with the actual tasks. Each subject works on two flat process models and two hierarchical ones. Group 1 starts with the flat representation of process model 1, while Group 2 works on the hierarchical representation of the same model. Subjects are confronted with hierarchical and flat models in an alternating manner. For each model, the subject is asked to “*explain roughly what the process describes.*” The exploratory study is concluded by a discussion with the subject to help reflecting on the study and providing us with feedback.

**Fig. 3.** Design of the exploratory study

Instrumentation. For each model, subjects received separate paper sheets showing the process models, allowing them to use a pencil for highlighting or taking notes, and juxtaposing the process models as desired. No written answers were required, only free talking. Audio and video recording are used as it has proven being useful for resolving unclear situations in think-aloud protocols [17].

4 Performing the Exploratory Study

Execution. The study was conducted in July 2012 in two locations. First, seven subjects participated at the University of Ulm, followed by two additional sessions at the University of Innsbruck, i.e., a total of nine subjects participated. To ensure that subjects were sufficiently familiar with declarative process modeling, they were provided with training material. Each session was organized as follows: First, the subject was welcomed and instructed to speak thoughts out loudly. To allow subjects to concentrate on their tasks, the sessions were performed in a “paper-workflow” manner, i.e., one supervisor was seated left to the subject, a second supervisor to the right. The sheets containing the study’s material were then passed from the left to the subject. As soon as the subject finished the task, the material was passed to the supervisor on the right. Meanwhile, the subject’s actions were audio- and video-recorded to gather any uttered thoughts.

Data Validation. In each session, only a single subject participated, allowing us to ensure that the study setup was obeyed. In addition, we screened whether subjects fitted the targeted profile, i.e., were familiar with process modeling and ConDec [6]. We asked questions regarding familiarity on process modeling, Con-Dec, and domain knowledge; note that the latter may significantly influence performance [18]. For this, we utilize a 7-point Likert Scale, ranging from “*Strongly agree*” (7) over “*Neutral*” (4) to “*Strongly disagree*” (1). Results are summarized in Table 4. Finally, we assessed the subjects’ professional background: all subjects indicated an academic background, i.e., were either PhD students or postdocs. We conclude that they had a profound background in process modeling (the least experienced subject had 2.5 years of modeling experience) and were moderately familiar with ConDec.

Table 4. Demographics (5–11 based on 7-point Likert Scale)

	Minimum	Maximum	Median
1) Years of modeling experience	2.5	7	5
2) Models read last year	10	250	40
3) Models created last year	5	100	10
4) Average number of activities	5	50	15
5) Familiarity ConDec	2	6	3
6) Confidence understanding ConDec	2	6	4
7) Confidence creating ConDec	2	6	4
8) Familiarity software development	4	7	6
9) Familiarity teaching	4	7	5
10) Familiarity electronic companies	1	6	2
11) Familiarity buying apartments	1	6	4

Data Analysis. Our research focuses on sense-making of declarative process models. On one hand, we investigate strategies applied by subjects in understanding process models, on the other, we explore typical phenomena and pitfalls in this process. For this purpose, data analysis comprised the following stages.

1. Transcription of the subjects' verbal utterances
2. Creation of graphs describing the order in which subjects mention activities
3. Analysis of transcripts using grounded theory

In (2), for each process model we create a graph representing the order activities were mentioned by the subjects. For this purpose, we utilize the transcripts created in (1), but also video recordings to identify when subjects visited an activity without talking about it. In (3), we apply grounded theory to the transcripts to explore and understand phenomena appearing when subjects make sense of declarative process models. As a starting point, transcripts are inspected, marking aspects that caused confusion, were misinterpreted or left out. In a second iteration, we revisit the marked areas and search for new aspects. This process of open coding analysis is repeated until no new aspects can be found. Afterwards, we perform axial coding, i.e., we repeatedly group aspects to form high level categories. We count the number of identified markings per category.

5 Findings

Based on the findings of our data analysis, we identified different ways how declarative models are read and interpreted.

5.1 Reading Declarative Business Process Models

When analyzing graphs and transcripts, we observed that subjects consistently adopted similar strategies when reading declarative models. For example, Fig. 4 shows the flat version of the first model and a typical strategy to understand that model. The model consists of two components. The first one contains activities “*receive bug report*” and “*search for bug in archive*”. The second component comprises all other activities. The dotted arrows display how three out of five subjects (Group 1) read the model to understand it.

Regardless of whether sub-processes were present or not, they described the process in the order activities were supposedly executed, i.e., tried to describe the process in a *sequential way*. Hence, as a first step, subjects skimmed over the process model to find an *entry point* where they could start with describing the (main) process: “...Ok, this is the first activity since it has this init constraint...” Interestingly, subjects appreciated when a clear starting point for their explanations could be found: “...it is nice that we have an init activity, so I can start with this...” Relating to the model depicted in Fig. 4, subjects started with “*receive bug report*” because of the init constraint. Then, they mentioned “*search for bug in archive*”. A declarative process model, however, does not necessarily have a unique entry point, apparently causing confusion:

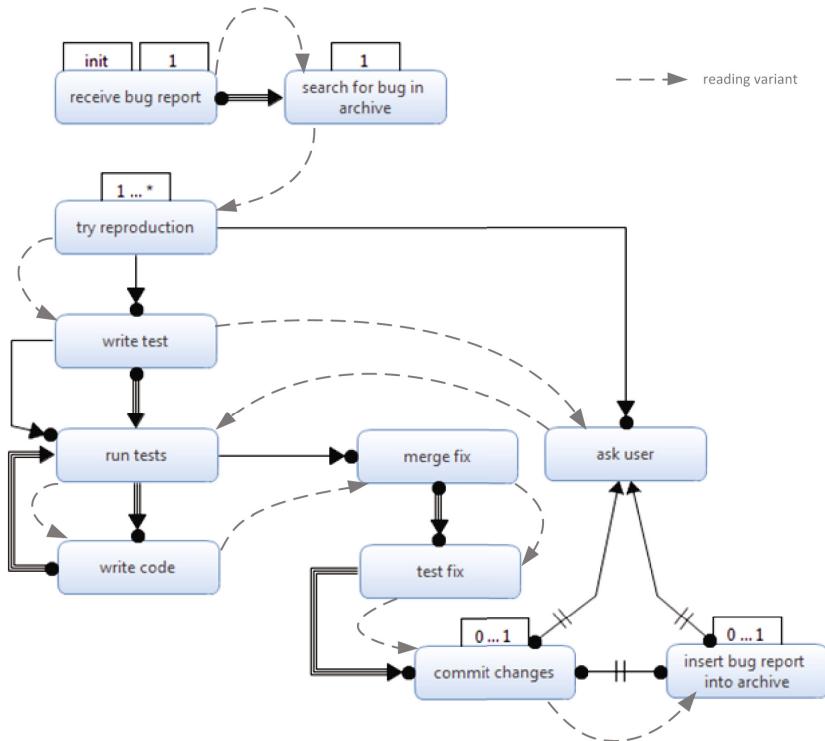


Fig. 4. First process model and a reading variant

“Well... gosh... I’ve got no clue where to start in this model...” The subjects used two different solutions for this kind of situation. Either they looked for a last constraint (“So, we don’t have init, but we have last...”) or they assumed the upper left corner of the model to be its entry point (“Ok... so first of all I have three initial I would say start activities...”). After having identified an entry point, subjects tried to figure out in which order activities are to be executed: “After given duties to the apprentices there should come these two tasks...”

This routine was iterative, i.e., if parts of a model were not connected, subjects applied the same strategy for each component, i.e., they started again at the upper left corner of these components. We observed this behavior independent of the respective process model or subject. Regarding our example (cf. Fig. 4), after describing the first component, subjects took a look at the second one. As there was no init constraint, they started in the upper left corner (“try reproduction”) and followed the other activities in a sequential way. Two subjects had problems, since there is no connection between the two parts of the model: “Ah, then there is a different process because these are not connected...” Likewise, a subject got irritated with single activities that had no connection to the rest of the model: “...ah this one, there’s no constraint here. You are just trying to confuse me.” Finally, subjects indicated where the process supposedly ends: “...the process

ends with the activity give lessons... ” When there was no last constraint (cf. Fig. 4), subjects stopped describing the process model after having mentioned all activities of all components.

If a model contained sub-processes (cf. Table 2), subjects preferred talking first about the main process in the above specified way before describing the sub-processes. When reading sub-processes the subjects used the same routine as for the main process, except two subjects. One of them described all and the second subject one out of four sub-processes completely *backwards*, i.e., following the semantics of precedence constraints, instead of describing them sequentially.

5.2 Single Building Blocks

Flat Declarative Process Models. In general, when subjects try to make sense of a model, they name activities and their connections. Sometimes, it happened that subjects missed single or small groups of activities. Regarding the first and second flat process model, no activities were left out. Three out of five subjects missed either 2, 4 and 17 activities in the flat version of the third process model (26 activities). The 17 activities were not referred to because a subject did not look at the components of the process model in detail. Four activities were not mentioned in the fourth flat process model: two out of four subjects did not mention one and three activities out of 26 activities. In summary, 27 out of 294 activities were missed in flat process models.

When describing a model sequentially, subjects name activities explicitly and most of the connections, i.e., the constraints, implicitly. However, most subjects did not mention existence constraints. This behavior could not be found for any other constraint. For 12 out of 18 models (9 subjects described two flat process models) subjects ignored one or more existence constraints. Table 5 shows the number of possible mentions of existence constraints per process model and the number of existence constraints that were ignored by the subjects. Summing up, subjects left out 34 of 78 existence constraints in flat process models.

Table 5. Existence constraints

Number of	Type	Proc. 1	Proc. 2	Proc. 3	Proc. 4
possible mentions of existence constraints	flat	25	8	5	40
	hierarchy	28	20	4	65
not mentioned existence constraints	flat	9	1	3	21
	hierarchy	19	2	1	30

Hierarchical Declarative Process Models. Regarding hierarchical process models, subjects tended to miss less activities. Two out of four subjects forgot to mention one activity in the first process model (11 activities). Regarding the second and third process model, no activities were left out. Three out of five subjects missed one activity in the hierarchical version of the fourth process model (23 activities). In summary, 5 out of 331 activities were missed in hierarchical process models.

Concerning the existence constraints in hierarchical process models, for 11 out of 18 models (9 subjects described two hierarchical process models), one or more existence constraints were not mentioned. As shown in Table 5, 52 from 117 existence constraints were ignored in hierarchical process models.

Flat and Hierarchical Declarative Process Models. As far as the interpretation of constraints is concerned, subjects had relatively little problems irrespective of whether the models were flat or hierarchical. As illustrated in Table 3, 12 different constraint types were used in the experimental material. To accomplish their task, subjects had cheat sheets available and could look constraints they did not know up. Except for the precedence constraint, which caused considerable difficulties, subjects faced no notable problems. Four out of nine subjects used the precedence constraint in a wrong way. According to Sect. 2, the definition of this constraint is that “*B can only be executed, if A has been executed before*”. The subjects used it the other way round, i.e., “*So if we perform receive incoming good [A] then do quality check [B] should be performed afterwards...*” One subject stated the absence of the precedence constraint between two components of a model: “*But still I don't get the relation between this part and the other one, so this is the problem, because I understand the flow, but I don't understand the relation between the two parts. Because there is no precedence.*” Additionally, the missing direction of the coexistence constraint caused one subject troubles: “*Let's see, so I would say you kinda start with these two activities, I'm not sure which one...*”

5.3 Combination of Constraints

Constraints between Two Activities. The first process model contained two and the fourth process model five situations where two constraints link two activities. In 6 out of these 7 cases, the direction of the constraint arrows are directly opposed to each other. For example, one needs to get offers for interior of an apartment before buying them (precedence constraint). After the interior is bought, it is not reasonable to get new offers (negation response). The subjects had no troubles to understand these situations. However, in the first process model there is a case where a precedence constraint and a chained response constraint link the two activities “*write test*” and “*run tests*”. Both arrows are pointing to the second activity (cf. Fig. 4). The precedence constraint ensures that before the first execution of “*run tests*”, “*write test*” must be executed at least once, i.e., it is not possible to run a test before it was written. The chained response constraint tells us that “*If A has been executed, B must be executed immediately afterwards.*”, meaning that after the test was written, it must be run directly afterwards; 4 out of 9 subjects had troubles with “the second arrow”, i.e., the precedence constraint. Two of them claimed that it is redundant (“*This part is redundant, right?*”), two even thought it is wrong (“*Over this relation, this is a precedence, so I think this is, ah, this can be removed.*”). The other 5 subjects ignored the precedence constraint.

Hidden Dependencies. Three out of the four process models contain hidden dependencies (cf. Sect. 2). Since these interactions are not *explicitly* visible, it is not sufficient that the analyst only relies on the information displayed explicitly, but must carefully examine the process model for these hidden dependencies as well. Our results show that the subjects mostly ignored hidden dependencies, i.e., only in 8 out of 36 models, a hidden dependency was mentioned or found: “*I have to execute prepare lesson in detail at least once, therefore, to fulfill the precedence constraint, I must execute prepare teaching sequence too.*”

6 Discussion

Reading Declarative Process Models. Subjects preferred describing process models in an iterative and sequential way. They started with the entry point of a component describing it in a sequential way and repeating this procedure for every component of the process model. The sequential way of describing models is surprising, as it is known that declarative process models rather convey circumstantial information (overall conditions that produce an outcome) than sequential information (how the outcome is achieved) [19]. In other words, in an imperative model, sequences are made explicit, e.g., through sequence flows. In a declarative process model, however, such information might not be available at all. As subjects tend to talk about declarative models in a sequential manner, it appears as if they prefer this kind of information. Interestingly, similar observations could be made in a case study on declarative process modeling [17]. Therein, sequential information, such as “*A before B*” or “*then C*” was preferred for communication.

Single Building Blocks. Regarding the interpretation of single building blocks, subjects mentioned activities and constraints when trying to understand the model. Overall, they had relatively little problems with the interpretation of single building blocks. Exceptions seem to be precedence and existence constraints. As a possible explanation these constraints are too simple and are thus not mentioned at all; further, cheat sheets are not used (cf. dual-process theory [20] describing the interplay of implicit unconscious and explicit controlled processes). Another explanation is that subjects were biased by previous knowledge about imperative models. Regarding the precedence constraint, it nearly looks like the arrow used in imperative process modeling notations.

Combining Constraints. The interplay of constraints seems to pose a challenge, especially hidden dependencies. One explanation could be that subjects simply forgot looking for them, as reading declarative models can quickly become too complex for humans to deal with [6]. As mentioned earlier, in 8 out of 36 models subjects found a hidden dependency. In 5 of these 8 cases, they were found in the second process model, which has the smallest number of activities, constraints and constraint types (cf. Table 2). This indicates that if a model is not too complex, subjects will be able to find hidden dependencies. Given this

finding, it seems plausible that the *automated* interpretation of constraints can lead to significant improvements regarding the understandability of declarative process models [21,5].

Differences between Flat and Hierarchical Process Models. Subjects did not distinguish between flat and hierarchical process models when reading the models. They used the same description strategy for components and sub-processes. Interestingly, subjects left out more activities in flat than in hierarchical process models (cf. Sect. 5.2). A reason for this phenomenon could be *abstraction* [22], i.e., hierarchy allows aggregating model information by hiding the internals of a sub-process using a complex activity. Thereby, information can be easier perceived. All other aspects we found could be observed in flat and hierarchical models equally.

Limitations. Our work has the following limitations. First, the number of subjects in the exploratory study is relatively low (9 subjects), hampering result generalization. Nevertheless, it is noteworthy that the sample size is not unusual for this kind of empirical investigation due to the substantial effort to be invested per subject [23]. Second, even though process models used in this study vary in the number of activities, number of constraints, and existence of sub-processes, it remains unclear whether results are applicable to declarative process models in general, e.g., more complex models. Third, all participating subjects indicated academic background, limiting result generalization. However, subjects indicated profound background in business process management, hence, we argue that they can be interpreted as proxies for professionals.

7 Related Work

We investigated the sense-making of declarative process models. The understanding of a declarative process model with respect to modularization has been investigated in [13]. However, opposed to our work, theory rather than empirical data is used for analysis. The role of understanding declarative process models during *modeling* has been investigated in [9]. Similar to our work, it has been postulated that declarative models are most beneficial when sequential information is directly available, as empirically validated in [17,5]. With respect to the understanding of process models in general, work dealing with the understandability of *imperative* business process models is related. The Guidelines of Modeling (GoM) describe various quality considerations for process models [24]. The so-called ‘Seven Process Modeling Guidelines’ (7PMG) accumulate the insights from various empirical studies, e.g., [25], to develop a set of actions a system analyst may want to undertake to avoid issues with respect to understandability [26]. The understandability of imperative process models is investigated empirically in [2]. As example of understandability issues in conceptual systems, [27] investigates if UML analysis diagrams increase system analysts’ understanding of a domain.

The impact of hierarchy on understandability has been studied in various conceptual modeling languages, such as imperative business process models [28], ER diagrams [29], and UML statechart diagrams [30] (an overview is presented in [22]). Still, none of these works deals with the impact of hierarchy on understandability in declarative process models.

While the effectiveness and usability of design guidelines for multiple diagrams were evaluated in [31], there are neither guidelines for designing nor for easily understanding declarative process models.

8 Summary and Outlook

Declarative approaches to business process modeling have recently attracted interest as they provide a high degree of flexibility [6]. However, the increase in flexibility comes at the cost of understandability, and hence might result in maintainability problems of respective process models [6,32,33]. The presented exploratory study investigates how subjects make sense of declarative business process models and provides insights into occurring problems. The results indicate that subjects read declarative process models in a sequential way. While single constraints caused only minor problems with exception of the precedence constraint, the combination of several constraints seems to be more challenging. More specifically, the subjects of this exploratory study mostly failed to identify hidden dependencies caused by combinations of constraints.

Even though the data we collected provided first insights into the process of understanding declarative models, further investigations are needed. Replications utilizing more complex models seem to be appropriate means for additional empirical tests. Although the think-aloud protocols already provide a detailed view on the reasoning processes of an analyst, we plan to employ eye movement analysis for more detailed analysis. The latter allows identifying areas, the analyst is focusing on in combination with insights on the required cognitive effort (similar to process modeling [34]). Based on these insights, we intend to evolve our work toward empirically founded guidelines enabling better understandability of declarative process models.

References

1. Mylopoulos, J.: Information modeling in the time of the revolution. *Information Systems* 23, 127–155 (1998)
2. Reijers, H.A., Mendling, J.: A Study into the Factors that Influence the Understandability of Business Process Models. *SMCA* 41, 449–462 (2011)
3. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer (2012)
4. Hildebrandt, T., Mukkamala, R., Slaats, T.: Nested dynamic condition response graphs. In: Proc. FSEN 2012, pp. 343–350 (2012)
5. Zugal, S., Pinggera, J., Weber, B.: The impact of testcases on the maintainability of declarative process models. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) *BPMDS 2011 and EMMSAD 2011. LNBP*, vol. 81, pp. 163–177. Springer, Heidelberg (2011)

6. Petic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, TU Eindhoven (2008)
7. Barba, I., Weber, B., Valle, C.D., Ramírez, A.J.: User Recommendations for the Optimized Execution of Business Processes. DKE (2013)
8. Zugal, S., Pinggera, J., Weber, B.: Assessing process models with cognitive psychology. In: Proc. EMISA 2011, pp. 177–182 (2011)
9. Zugal, S., Pinggera, J., Weber, B.: Toward Enhanced Life-Cycle Support for Declarative Processes. Journal of Software: Evolution and Process 24, 285–302 (2012)
10. Ericsson, K.A., Simon, H.A.: Protocol analysis: Verbal reports as data. MIT Press (1993)
11. van der Aalst, W.M.P., Petic, M.: DecSerFlow: Towards a truly declarative service flow language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
12. Parnas, D.L.: On the Criteria to be Used in Decomposing Systems into Modules. Communications of the ACM 15, 1053–1058 (1972)
13. Zugal, S., Soffer, P., Pinggera, J., Weber, B.: Expressiveness and Understandability Considerations of Hierarchy in Declarative Business Process Models. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (eds.) BPMDS 2012 and EMMSAD 2012. LNBIP, vol. 113, pp. 167–181. Springer, Heidelberg (2012)
14. Bassey, M.: Case study research in educational settings. Doing qualitative research in educational settings. Open University Press (1999)
15. Corbin, J., Strauss, A.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. SAGE Publications (2007)
16. Haisjackl, C.: Test Driven Modeling meets Declarative Process Modeling – A Case Study. Master's thesis, University of Innsbruck (2012)
17. Zugal, S., Haisjackl, C., Pinggera, J., Weber, B.: Empirical Evaluation of Test Driven Modeling. Accepted at the International Journal of Information System Modeling and Design (2012) (to appear)
18. Khatri, V., Vessey, I., Ramesh, P.C.V., Park, S.J.: Understanding Conceptual Schemas: Exploring the Role of Application and IS Domain Knowledge. Information Systems Research 17, 81–99 (2006)
19. Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative versus Imperative Process Modeling Languages: The Issue of Understandability. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009. LNBIP, vol. 29, pp. 353–366. Springer, Heidelberg (2009)
20. Kahneman, D.: Maps of bounded rationality: A perspective on intuitive judgment and choice. Nobel Prize Lecture 8, 449–489 (2002)
21. Zugal, S., Pinggera, J., Reijers, H., Reichert, M., Weber, B.: Making the Case for Measuring Mental Effort. In: Proc. EESSMod 2012, pp. 37–42 (2012)
22. Zugal, S., Pinggera, J., Mendling, J., Reijers, H., Weber, B.: Assessing the Impact of Hierarchy on Model Understandability-A Cognitive Perspective. In: Proc. EESSMod 2011, pp. 123–133 (2011)
23. Costain, G.F.: Cognitive Support During Object-oriented Software Development: The Case of UML Diagrams. PhD thesis, University of Auckland (2007)
24. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of Business Process Modeling. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 30–49. Springer, Heidelberg (2000)

25. Mendling, J., Verbeek, H., van Dongen, B., van der Aalst, W., Neumann, G.: Detection and prediction of errors in epcs of the sap reference model. *DKE* 64, 312–329 (2008)
26. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). *Information & Software Technology* 52, 127–136 (2010)
27. Burton-Jones, A., Meso, P.N.: Conceptualizing systems for understanding: An empirical test of decomposition principles in object-oriented analysis. *Information Systems Research* 17, 38–60 (2006)
28. Reijers, H., Mendling, J., Dijkman, R.: Human and automatic modularizations of process models to enhance their comprehension. *Inf. Systems* 36, 881–897 (2011)
29. Moody, D.L.: Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) *ADBIS 2004. LNCS*, vol. 3255, pp. 129–143. Springer, Heidelberg (2004)
30. Cruz-Lemus, J.A., Genero, M., Morasca, S., Piattini, M.: Using Practitioners for Assessing the Understandability of UML Statechart Diagrams with Composite States. In: Hainaut, J.-L., Rundensteiner, E.A., Kirchberg, M., Bertolotto, M., Brochhausen, M., Chen, Y.-P.P., Cherfi, S.S.-S., Doerr, M., Han, H., Hartmann, S., Parsons, J., Poels, G., Rolland, C., Trujillo, J., Yu, E., Zimányie, E. (eds.) *ER Workshops 2007. LNCS*, vol. 4802, pp. 213–222. Springer, Heidelberg (2007)
31. Kim, J., Hahn, J., Hahn, H.: How do we understand a system with (so) many diagrams? cognitive integration processes in diagrammatic reasoning. *Information Systems Research* 11, 284–303 (2000)
32. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The Declarative Approach to Business Process Execution: An Empirical Test. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) *CAiSE 2009. LNCS*, vol. 5565, pp. 470–485. Springer, Heidelberg (2009)
33. Zugal, S., Pinggera, J., Weber, B.: Toward Enhanced Life-Cycle Support for Declarative Processes. *Journal of Software: Evolution and Process* 24, 285–302 (2012)
34. Pinggera, J., Furtner, M., Martini, M., Sachse, P., Reiter, K., Zugal, S., Weber, B.: Investigating the Process of Process Modeling with Eye Movement Analysis. In: La Rosa, M., Soffer, P. (eds.) *BPM Workshops 2012. LNBIP*, vol. 132, pp. 438–450. Springer, Heidelberg (2013)

Blending BPMS with Social Software for Knowledge-Intense Work: Research Issues

Nancy Alexopoulou¹, Mara Nikolaïdou², and Christian Stary¹

¹ Johannes Kepler University,

Department of Business Information Systems – Communications Engineering,
Linz, Austria

² Harokopio University of Athens, Department of Informatics and Telematics,
Athens, Greece

Abstract. Knowledge-intense processes are by their very nature exploratory, non-repetitive in detail and not completely known in advance. Flexibility, effective knowledge management and efficient collaboration are important requirements of such processes. A typical flow-oriented BPMS, relying on the generation of a model a priori and imposing a specific sequence of tasks to process participants is not appropriate for such processes, as it does not align with their nature and cannot satisfy their requirements. Therefore, alternative approaches are explored by the research community. An emerging trend towards this direction is the incorporation of social software features in BPMS. However, bringing the BPMS and social software together is not a straightforward task in the context of knowledge-intense work. Several crucial issues arise that should be closely investigated for an appropriate approach to be developed ensuring an efficient execution of knowledge-intense processes. In this paper, a number of such issues are identified helping towards the detection of an effective and efficient solution when blending features from both software types.

Keywords: knowledge-intense business processes, BPMS, social software.

1 Introduction

The flow-oriented paradigm constitutes the dominant business process modeling approach adopted by the majority of current BPMS (Business Process Management System) [1] products. According to this paradigm, business process tasks are orchestrated in a specific sequence using a modeling language such as BPMN (Business Process Modeling Notation) [2], which is subsequently interpreted and enacted by the process engine being part of the BPMS. While typical flow-oriented approaches effectively support the needs of well-structured processes, i.e. processes with well-defined steps, they fall short however to satisfy flexibility requirements addressed in ill-defined processes [3]. The former rely on the generation of a model a priori, which is then enacted multiple times. This logic aligns with their standardized

repetitive type of work. In contrast to that, ill-defined processes are by their very nature exploratory, non-repetitive in detail and not completely known in advance. The main concern in the latter is communication and knowledge-sharing among participants, rather than the coordination of activities [4]. Thus, they are often referred to as collaborative or knowledge-intense processes [5]. A process is regarded knowledge-intense if its value can only be created through the fulfillment of knowledge requirements of process participants. Flexibility is an inherent requirement of such processes [6], as the workflow during enactment is mainly determined by decisions made, often on the fly, by knowledgeable actors, and hence cannot be prescribed by a BPMS. In particular, *flexibility, efficient collaboration and effective knowledge management* are key requirements for knowledge-intense processes ([7], [8], [9]) that cannot be effectively served by classical BPMSs.

In order to meet the requirements of knowledge-intense processes, researchers are investigating and developing alternative approaches, such as KMDL [10] and CommonKADS [11], focusing on representation, modeling and analysis of knowledge-intense processes and POKM [12], which is a method for capturing the expert's knowledge in such processes. Initiatives focusing on the efficient execution of knowledge-intense processes include efforts such as KnowMore [13], FRODO [14] and PROMOTE [15]. KnowMore augments knowledge-intense tasks of a business process with recommendations and decision support information. FRODO addresses the issue of flexibility, adopting weakly structured workflows, to enable interleaved modeling and execution of knowledge-intense processes. PROMOTE extends the more general method of typical Business Process Management (BPM) with strategic decision, knowledge process discovery, organizational memory creation and enterprise knowledge evaluation.

While constituting important contributions to the field of knowledge-intense process support, none of these initiatives emphasizes the aspects of efficient collaboration between participants through various means as well as the externalization of participants' tacit knowledge [16], which the knowledge residing in peoples' heads. This explains the growing interest towards the emerging trend reflected upon the established term *Social BPM* [17]. The purpose of social BPM is the adoption of social software features in the BPM (Business Process Management) discipline in order to foster flexibility, knowledge management and inter-participant collaboration in business process support. Such merits are of paramount importance in today's highly dynamic market environments, which drive work in modern enterprises to increasingly become more and more knowledge-intense, addressed for example by initiatives such as Enterprise 2.0 ([18], [19]).

Social software is rapidly gaining acceptance as revealed by a new generation that is accustomed to use platforms such as Facebook (www.facebook.com) for communication and socialization purposes. This new way of communication is quickly spreading to the business life as well through the evolution of informal business networks such as LinkedIn (www.linkedin.com). Utilizing social characteristics in BPM could prove worthwhile for several reasons, especially for knowledge-intense processes, since they could effectively serve their key requirements as discussed in the following:

- Communication and collaboration are key features, effectively supported by social software ([20], [21]). Such functionalities are of great significance for knowledge-intense processes, which rely more on participant collaboration rather than an engine enforcing the steps that should be followed.
- Social software can connect actors easier to the resources they actually need [22]. More specifically, they have the ability to find, learn about and connect with the right people, information and other resources to deal with unanticipated situations, thus promoting process flexibility.
- Social software facilitates knowledge updating as it allows for tacit knowledge to be easily externalized and shared in a way that new knowledge is created, by enabling users to reach out to a large number of relevant participants and engage in discussions, and by capturing and making searchable such informal discussions [21].

As the beneficial influence of social characteristics in BPM has already been discerned by the research community, there are already research initiatives towards this direction [23]. However, bringing the BPMS and social software together is not a straightforward task when supporting knowledge-intense work. Important issues arise that should be closely investigated for an appropriate approach to be developed that can effectively satisfy the aforementioned requirements of knowledge-intense processes.

The purpose of this paper is to address research issues emerging from infusing social software features in a BPMS environment, targeting the efficient execution of knowledge-intense processes. To identify such issues, we were grounded on the business process modeling perspectives proposed by Curtis et al [24]. Issues regarding the infusion of social software in BPMS are then discussed from each perspective. Specifically, the paper is organized as follows. Section 2 presents the state of the art regarding the adoption of social features in BPM. Research issues are examined from each perspective in section 3. Conclusions are given in section 4.

2 State of the Art Regarding the Adoption of Social Features in BPM

Business Process Management utilizing social networking concepts has recently gained momentum, due to social software characteristics like weak ties and mutual service provision, which fulfill requirements of collaborative environments ([23], [25]). Moreover, research community's intense interest is reflected upon related conferences and workshops that identify emerging issues, such as the International Workshop Series on Business Process Management and Social Software (<http://www.bpm2.org/>).

Literature is rich in contributions concerning the adoption of social software features in the BPM discipline. A part of this research focuses on how social software can be used to support collaborative business process design ([26], [27], [28]). Koschmider et al., for example, suggest in [28] an approach, according to which process models can be shared and exchanged based on the network proximity of modelers. Other approaches focus on using social tagging mechanisms for relating

models dynamically [29] or managing them in a model repository [30]. Brambilla et al. [31] have proposed a notation for social BPM defined as a BPMN 2.0 extension. It enables the annotation of specific tasks as collaborative ones and their potential execution within a social network environment. In [32] a BPM infrastructure bearing social software features is proposed, targeting both collaborative modeling as well as business process execution in a fashion that mashes up definition and operation of business processes. The corresponding tool, called AGILIPo, is currently under development and testing. In [33] the authors examine how the architectural principles behind BPMS and social software can be combined in order to develop a unified infrastructure supporting features of both software types. Johannesson et al. [34] suggest a set of guidelines for augmenting BPMS with social software features, which may be effective for knowledge-intense process modeling, though the execution model is not clearly defined. Based on the aforementioned efforts it becomes evident that knowledge-intense process execution will benefit from the integration of social software features in BPMS environments. Thus, we explore a taxonomy of the issues to be solved towards the development of such environments, facilitating the effective execution of knowledge-intense processes.

In practice, though the phenomenon of social networking within an organization gains momentum, as investigated by Richter and Riemer in [35], its usage is restricted in communication and information sharing. That is, the social software infrastructure is used only for exchanging information or performing trivial tasks, such as arranging a meeting, and not for integrated BPM solutions, which seems to be the step forward.

3 Issues Emerging from the Infusion of Social Software Features in BPMS

To identify challenges posed by the infusion of social software features into BPMS, attributes of both software types should be comprehensively explored. To this end, the business process modeling perspectives proposed by Curtis et al. [24] for executable business process description can be applied. According to Curtis et al. [24], a business process model can be viewed from a functional, behavioral, organizational and informational perspective. The functional perspective depicts what activities are performed. When and how activities are performed constitutes the behavioral perspective, while where and by whom they are executed corresponds to the organizational perspective. What information entities are created and processed during each activity is examined in the informational perspective.

Table 1 juxtaposes characteristics of these two software types distinguished in the aforementioned perspectives. As BPMS and social software have a different orientation, they reasonably bear different features, which can even be regarded to a large extent contradictory, as shown in Table 1. Taking into account the diverse features of social software and BPMS, the main problem arising is how BPMS and social software could be merged to efficiently support the execution of knowledge-intense business processes. This question will be further elaborated in the following by examining each perspective separately, having in mind the specific characteristics of each software type .

Table 1. Juxtaposing features of social and BPM software from four business process modeling perspectives

Business Process Modeling Perspectives	Social Software	BPMS
Functional Perspective	<ul style="list-style-type: none"> - social-specific activities <ul style="list-style-type: none"> - profile management - networking - communication - context creation - searching 	<ul style="list-style-type: none"> - business-specific activities (tasks)
Behavioural Perspective	<ul style="list-style-type: none"> - wisdom of the crowds - social interaction - social production 	<ul style="list-style-type: none"> - wisdom of the expert - prescribed task execution - predefined input from each participant
Organizational Perspective	<ul style="list-style-type: none"> - egalitarianism - weak ties - public access 	<ul style="list-style-type: none"> - role hierarchy - strong ties - access policies specified by top management
Informational Perspective	<ul style="list-style-type: none"> - content or context information concerning artifacts or physical objects 	<ul style="list-style-type: none"> - business or physical objects

Functional Perspective

Functionality of a business process is described through business-specific activities often called tasks [2], although a hierarchical relationship may also be defined between these two terms. A business activity can be anything performed within the context of a specific business process. However there are strict descriptions of their input and output as well as the roles/participants responsible for their execution, which constitute parts of their definition. Activities supported in social software, on the other hand, have a more narrowed scope. Thus, we group them in five main categories, namely, profile management, networking, communication, searching and context creation. The first category comprises activities such as profile creation, update, view, etc., while networking involves creating and/or participating in social links or groups. The third category involves message exchange, announcement posting and forum initiation and participation. Searching activities for extracting a wide variety of information (people, places, photos, jobs, etc.) are also provided by

social software. Lastly, another important type of activities offered by such software are those concerning the creation of context, i.e. metadata for the existing data. This can be accomplished through tagging (i.e. using keywords to classify data), evaluating (e.g. through rating or endorsing) and annotating. It should be noted that any participant of a social network or wiki may perform any activity without restrictions.

Would there be any benefit for knowledge-intense processes in case BPMS specifically supported such activities and provided the means for their explicit description, as suggested in [31]? Adopting the idea of creating a profile for each employee within the organization might help, along with corresponding searching and annotation capabilities, towards identifying the appropriate person, in terms of knowledge and experience, for the solution of a problem or the execution of a task in general. In addition, as time in the execution of an organizational process is crucial, the BPMS should also be able to detect the availability of each participant and make suggestions based on participant's current workload as indicated by Bessai and Nurcan [52]. However, their involvement could not be considered only on a voluntary basis, following the social computing model. What kind of information should be included in employees' profiles within the context of an organization to serve the needs of process execution is an issue requiring further research. Information indicating the employees' position, role and responsibilities within the limits of a specific organization should also be included, since they are related to the potential execution of specific tasks, according to predefined business rules.

Apparently, networking and communication activities, a key feature of social computing, may enhance collaboration between participants and help them in knowledge sharing. To this end, it may be useful to augment BPMS with capabilities such as announcement posting and forum initiation as well as creation and participation in social groups. For example, a participant may invite friends, colleges or intermediates – e.g. participants belonging to a specific group - to help him/her to complete a specific task [36]. However, how exactly such features could contribute to the efficiency in the execution of knowledge-intense processes and under which conditions, should be more concretely examined.

Lastly, the concept of context creation could also be adopted in BPM software. According to Erol et al. [4], for example, attaching personal user-oriented description of resources to tasks could create an analogy between tasks and tags in folksonomies. Task-folksonomies, as referred in [4], could be considered an informal way of representing process knowledge, as opposed to the formal representation used in conventional business process models. It is worth investigating how task-related folksonomies, created by participants themselves, could be used to enable automatic detection of resources (data, people, etc.) required for the execution of specific task. Furthermore, the way a specific participant or group of participants may combine tasks to complete a specific goal could be considered as context creation and utilized by other participant with similar goals. Overall, the entire concept of activity/task modeling in a combined social and BPMS environment may need to be reconsidered.

Behavioral Perspective

Two fundamental features of social software are social interaction and social production. The first concerns the communication between individuals without predefined rules (e.g. Facebook), while the second is about the creation of artifacts by

combining the input from independent contributors without a-priori specification of the way doing this (e.g. Wikipedia). In contrast, using a typical BPMS, the interactions among participants are prescribed through rigid process models, specifying the order of tasks as well as the exact way each participant is involved, so that a certain business goal is reached. This mode of work might not suit knowledge workers, as they need support for creative problem solving rather than constraints set by a software system. Could the incorporation of social features in BPM software offer a way of working suitable for knowledgeable actors operating within a specific business environment? What would then be the meaning of social interaction and social production in such an environment? Knowledgeable actors may work together to reach a specific goal in a similar fashion as individual contributors combine input in Wikipedia. Should their interaction be free from any constrained or rule? In other words, the way social interaction and social production notions should be interpreted in an organizational context needs to be explored.

Even when a participant is not knowledgeable in the sense that he/she does not make decisions but mainly performs procedural tasks, he/she might prefer not to work according to the strict workflow-oriented fashion imposed by a conventional BPMS. If workers do not follow a prescriptive model and are let to perform individual processes, they might express new ideas and make suggestions for process improvement, getting thus actively involved in the development of new business process patterns. Furthermore, “collective intelligence” [4] of many people may lead more effectively to the solution of a problem than the knowledge of an expert who is sometimes difficult to be identified. This adheres to the “wisdom of the crowds” idea introduced in [37]. However, one could argue that enforcing participants to follow strict business process models, offers a “safer” way to execute a process with regard to a business goal as in this way it is ensured that participants know exactly what to do and when to do it and that they will definitely contribute in the completion of a process. Indeed, proactive contribution cannot be taken for granted. Participants may not contribute if they are not obliged to. Thus, the issue that arises is how to combine social software features, promoting creativity and innovation, with BPMS support features that accommodate the required control over the executed processes, so that advantages from both sides can be exploited. Also, it should be stressed that for BPM to reap the fruits of social software, it should be ensured that participants are motivated for a proactive contribution [25]. Identifying such motivation mechanisms constitutes therefore another research matter.

If the combination of social with BPM software introduces a novel way of working in a business environment, what kind of process modeling approaches would be appropriate for describing such a way of working? In other words, what does the introduction of social technology into BPMS entail for business process design? Typical flow-oriented models as those described using for example BPMN [2] seem totally inappropriate. Could then message-oriented approaches, such as S-BPM [38], or perhaps an adapted version of them be more suitable? Or would it be better to adopt a goal-driven approach [39]? Or could hybrid approaches amalgamating diverse modeling paradigms be more promising?

Taking into account the current logic underlying social software, the definition of a complete process model prior to enactment would probably not make sense. The effective support of knowledge-based processes underlies an ill-structured model.

One could research whether there would be any meaning in specifying a kind of model to be followed during enactment or perhaps a model would not need to exist at all at the initial execution of a process [36]. Rather, it could be extrapolated automatically [40] through process mining techniques [41] based on worker's actions. In this respect, it might be more useful to model on a more fine-grained level, considering process constituents, i.e. tasks, as autonomous entities [42] and specifying for each task the associated resources and the respective roles allowed to execute it. The extracted process models can be subsequently analyzed and optimized. Through such an analysis stakeholders may gain insight into their everyday work and identify best practices. The identified process patterns could be made available for being shared among participants. These patterns may be continually updated based on the experience and knowledge of workers. Should such business process patterns be used together with a recommendation mechanism or would it be mandatory to follow a specific pattern for the realization of a specific business goal? Behind this dilemma, lies again the issue of ideally combining the freedom and proactive operation of participants offered by social software with conformance to business policies and rules ensured by BPMS. Ultimately, what would be the role of a workflow engine in such type of hybrid software? Would it be required or valid at all? Or could alternative technologies, such as shared spaces [43] turn out more appropriate? Lastly, can we stick to the typical business process lifecycle paradigm as we currently know it or do we need to reconsider it?

Organizational Perspective

Weak ties are formulated through social networks, as opposed to strong ties which are developed through relationships based on hierarchy and team structure. As indicated in [25], weak ties are spontaneously established contacts invoked not by management but by individuals. Egalitarianism [25] is about giving all participants the same rights to contribute, in contrast to organizational environments, where well-defined roles and role interrelationships determine responsibilities within the context of the organization, which in turn are depicted within BPMS environment. Access to information is also determined by roles and policies specified by top management, while social software environments allow for a wider access to information.

In any case, the responsibilities of each employee are determined by his/her position in the organization, accommodated by predefined responsibilities, according to well-established policies. Even in knowledge-intense processes, which should be executed within the context of an organization, where there are not prescribed steps to be followed, knowledgeable participants should adhere to business rules prescribing an organization's policies.

The way relationships are cultivated among business process participants through a social network, according to the aforementioned social software attributes, better facilitate and encourage the exchange of views and ideas [25]. As a result, externalization of tacit knowledge as well as sharing and dissemination of knowledge, which are key requirements for knowledge-intense processes, are better served through social software. However, in an organizational environment not anybody can do anything at anytime. For a harmonious operation of the enterprise, participants should comply with the policies and business rules holding within the enterprise. As a result, the appropriate balance between the freedom offered by social software

facilitating collaboration and knowledge diffusion on one hand, and organizational-specific policies reflected upon business process models and effectively supported by BPMS on the other hand, should be investigated. Such a balance should provide for maximum flexibility, ensuring at the same time that chaos is prevented. This entails that enterprise may need to change the established rules and policies towards Enterprise 2.0 concepts [18].

It should be stressed that the concept of role is key in organizational environments and therefore constitutes a fundamental entity in business process modeling [44]. Since such concept is missing in social software in terms of functional entities, the latter cannot adequately support organizational requirements in terms of business process execution. In a BPMS environment, the concept of “role” is used to denote a set of responsibilities within an enterprise that can be assigned to a specific actor category (e.g. a doctor). The integration of BPMS with social software might prompt for a reconsideration of role description. For example, as stated in [45], the concept of role should be assigned richer semantics to accommodate human interactions in knowledge-intense processes. This might mean to describe for each role the required knowledge and skills to obtain it. On the other hand, there are efforts, identifying the need to introduce the concept of the role, with loose semantics, in private social networks built to accommodate collaborative communities within the context of an organization [46].

To conclude, as also stated by Harrison-Broninski [45], better techniques are required for modeling relationships, both on a personal basis as users and within a process context as roles, in order to support human behaviors such as learning, adaptation and conflict resolution, and typical process features like goals, responsibilities and delegation of authority, which might serve to effectively perform knowledge-intense processes.

Informational Perspective

Information in social software regards objects like photos, songs, e-books etc. associated with metadata developed by participants using tagging, evaluating and annotating (see above). Utilizing the “wisdom of the crowds”, participants may also classify information formulating folksonomies, which may help others to seek the information they needed. Thus, context information is available for the content created by participants. In contrast, information in BPMS is depicted onto business objects such as order forms, receipts, invoices, etc., which are strictly related to activities as input or output data. Metadata are critical for knowledge-intense processes, as they correspond to an essential part of knowledge. Embodying therefore metadata to characterize the raw data and exploring how creation and sharing of metadata can be supported in BPMS, is of paramount importance for the promotion of knowledge elicitation.

Currently, modeling of such metadata is not supported in existing business process modeling languages. However, there are several research initiatives towards “context modeling” as it is called [47], falling into various scientific fields such as mobile computing, knowledge management, etc. Regarding the BPM field, there are also related research efforts aiming at capturing through context data the situation under which specific activities are performed, so that contingencies can be effectively handled ([48], [49]). Metadata or context modeling remains an open issue.

Table 2. Issues arising from the combination of BPMS with social software for supporting knowledge-intense processes

Business Process Modeling Perspectives	Issues
Functional Perspective	<ul style="list-style-type: none"> - Would it be beneficiary for the execution of knowledge-intense processes to create and maintain a profile for each employee? If yes, what kind of information should be included in such a profile? Who would be responsible for updating it? - How could networking and communication activities contribute to the efficient execution of knowledge-intense processes? - How could context created by participants may be used by others to enable automatic detection of the resources required for the execution of a specific task? - What does the integration of BPMS with social software entail for the concept of activity/task modeling?
Behavioral Perspective	<ul style="list-style-type: none"> - How should the notions of social interaction and social production be interpreted in a BPMS environment? - How could social software features, promoting participants' creativity and innovation be integrated with typical BPMS controlled flow of the executed processes, so that advantages from both sides are exploited? - What kind of motivation mechanisms should be established to encourage proactive contribution of process participants, even if they are not obliged to do so? - What kind of process modeling approach would be appropriate for describing the alternative way of working for process participants, arising from blending BPMS with social software? Would it be meaningful to execute process patterns to some extend or merely use them for recommendation purposes? - How should existing technologies from both software types be combined for the development of a hybrid system that ensures flexibility and enables effective knowledge management and efficient collaboration? Would the existence of a workflow engine be of any value for such a system? - Would business process lifecycle holds as is or a reconsideration of it would be necessary?
Organizational Perspective	<ul style="list-style-type: none"> - How should the structure of the organization (e.g. positions and responsibilities, policies) be represented within the environment supporting knowledge-intense processes? - What is the degree of freedom that actors should have in the execution of a business process? - What are the implications of combining BPMS with social software for the concept of role? Should it be modeled in an alternative way to accommodate additional semantics?
Informational Perspective	<ul style="list-style-type: none"> - How could the creation and sharing of metadata as part of folksonomies be supported in a BPMS environment in order to promote knowledge elicitation during process enactment? - Would it be useful for knowledge-intense processes to loosen the semantics of activity input and output? - How could the experience of participants fulfilling a goal or participating in completing a process instance be transformed to knowledge available to others having similar needs? - How can content quality, trust and reliability be ensured in a hybrid environment adopting features from both social software and BPMS?

Furthermore, one should consider omitting or reducing the semantics of data serving as activity input or output. Since information may be classified based on an ontology or folksonomy within a specific context, such as scientific field, administrative domain or language [50], a knowledgeable actor may identify semantic similarities between them, loosening the strict relation between activities input and output dictated by typical BPMS environments. Such a feature might promote flexibility in the way activities can be composed to meet a specific objective.

In social computing, participants receive recommendations, based on other participants' actions or experience [46]. When the actor is not obliged to follow a strict flow of activities, the experience of others may help in choosing what to do. In such cases, evident in knowledge-intense processes, the BPMS environment should provide for the transformation of past participants' recorded experience to knowledge available to those facing the same or similar situations [40].

Finally, it should be stressed that the uncontrolled creation of content from the social media has its own issues of trust, content quality and reliability that should be taken into account when considering it in a business environment [51].

Table 2 summarizes the identified issues for each perspective.

4 Conclusions – Future Work

Embedding social software features in BPMS seems promising for effectively supporting knowledge-intense processes. However, this amalgamation raises several important issues. In order to identify and address such issues, we structured our findings according to the modeling perspectives proposed by Curtis et al. [24]. These perspectives were used to juxtapose attributes from both software types. From the analysis conducted in this paper, it is ensued that the solution to effectively support knowledge-intense process execution within an organizational environment, might lie somewhere in-between the two software types, blending features from both of them. Coming up with an effective and efficient solution is a major research issue, presumably affected by process- and enterprise-specific parameters. In other words, the solution may vary depending on the characteristics of the business process and the enterprise in concern. The list of the identified issues is not exhaustive. However, we believe that examining the integration of social software with BPMS in respect to the specific modeling perspectives can help researchers in further identifying related issues, so that they can contribute to the incarnation of an effective solution for efficiently supporting knowledge-intense processes.

Future work targets at the prototypical development of a hybrid environment integrating social computing features into a BPMS engine for the efficient execution of knowledge-intense processes. Given the issues one should consider, a first step towards this direction is the identification of the appropriate modeling approach to activity execution, the integration of networking and communication features to co-ordinate activity executions and the introduction of participant profiles to describe the characteristics of knowledgeable actors.

Acknowledgments. The research leading to these results has received funding from the European Commission within the Marie Curie Industry and Academia Partnerships & Pathways (IAPP) programme under grant agreement n 286083.

References

1. Dumas, M., Aalst, W., Hofstede, A.: *Process-Aware Information Systems*. John Wiley & Sons Inc. (2005)
2. OMG, Business Process Management Notation. Version 2.0 (2011)
3. Nurcan, S., Edme, M.-H.: Intention-driven modeling for flexible workflow applications. *Software Process: Improvement and Practice* 10(4), 363–377 (2005)
4. Erol, S., Granitzer, M., Happ, S., Jantunen, S., Jennings, B., Johannesson, P., Koschmider, A., Nurcan, S., Rossi, D., Schmidt, R.: Combining BPM and social software: contradiction or chance? *Journal of Software Maintenance and Evolution: Research and Practice* 22(6-7), 449–476 (2010)
5. Gronau, N., Weber, E.: Management of Knowledge Intensive Business Processes. In: Desel, J., Pernici, B., Weske, M. (eds.) *BPM 2004*. LNCS, vol. 3080, pp. 163–178. Springer, Heidelberg (2004)
6. Swenson, K.: *Mastering the Unpredictable*. Meghan-Kiffer Press (2010)
7. Franca, B.S.J., Baiao, F.A., Santoro, M.F.: Towards characterizing Knowledge Intensive Processes. In: *Proceedings of the IEEE 16th International Conference on Computer Supported Cooperative Work in Design*, May 23-25 (2012)
8. Scheithauer, G., Hellmann, S.: Analysis and Documentation of Knowledge-Intensive Processes. In: La Rosa, M., Soffer, P. (eds.) *BPM Workshops 2012*. LNBIP, vol. 132, pp. 3–11. Springer, Heidelberg (2013)
9. Mundbrod, N., Kolb, J., Reichert, M.: Towards a System Support of Collaborative Knowledge Work. In: La Rosa, M., Soffer, P. (eds.) *BPM Workshops 2012*. LNBIP, vol. 132, pp. 31–42. Springer, Heidelberg (2013)
10. Gronau, N., Müller, C., Korf, R.: KMDL - Capturing, Analysing and Improving Knowledge-Intensive Business Processes. *J. UCS* 11(4), 452–472 (2005)
11. Schreiber, G., Akkermans, R., Anjewierden, A., Hoog, R., Shadbolt, N., DeVelde, W.V., Wielinga, B.: *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press, Cambridge (2002)
12. Trier, M., Müller, C.: Towards a Systematic Approach for Capturing Knowledge-Intensive Business Processes. In: Karagiannis, D., Reimer, U. (eds.) *PAKM 2004*. LNCS (LNAI), vol. 3336, pp. 239–250. Springer, Heidelberg (2004)
13. Abecker, A., Bernardi, A., Hinkelmann, K., Kuhn, O., Sintek, M.: Context-Aware, Proactive Delivery of Task-Specific Information: The KnowMore Project. *Information Systems Frontiers* 2(3-4), 253–276 (2000)
14. Aschoff, F.-R., Bernardi, A., Schwarz, S.: Weakly-structured workflows for knowledge-intensive tasks: an experimental evaluation. In: *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE 2003, pp. 340–345 (2003)
15. Hinkelmann, K., Karagiannis, D., Telesko, R.: PROMOTE. Methodologie und Werkzeuge für geschäftsprozessorientierten Wissensmanagement. In: Abecker, A., Hinkelmann, K., Maus, H., Müller, H.J. (eds.) *Geschäftsprozessorientiertes Wissensmanagement: Effektive Wissensnutzung bei der Planung und Umsetzung von Geschäftsprozessen*, pp. 65–90. Springer, Heidelberg (2002) (in German)
16. Polanyi, M.: *Personal Knowledge - Towards a Post-Critical Philosophy*. The University of Chicago Press, Chicago (1958)
17. Swenson, K., Palmer, N., Kemsley, S., Harrison-Broninski, K., Pucher, M., Das, M.: *Social BPM. BPM and Workflow Handbook Series*. CreateSpace Independent Publishing Platform (2011)

18. Kakizawa, Y.: In-house use of web 2.0: Enterprise 2.0. *NEC Technical Journal* (2007)
19. Enterprise 2.0 conference. *Enterprise 2.0: What, Why and How.* White paper (2009)
20. Avram, G.: At the Crossroads of Knowledge Management and Social Software. *The Electronic Journal of Knowledge Management* 4(1), 1–10 (2006)
21. Richardson, C.: Social Technologies Will Drive the Next Wave of BPM Suites. *Forrester Blogs* (2009), <http://blogs.forrester.com>
22. Dengler, F., Koschmider, A., Oberweis, A., Zhang, H.: Social Software for Coordination of Collaborative Process Activities. In: zur Muehlen, M., Su, J. (eds.) *BPM 2010 Workshops.* LNBIP, vol. 66, pp. 396–407. Springer, Heidelberg (2011)
23. Schmidt, R., Nurcan, S.: BPM and Social Software. In: Ardagna, D., Mecella, M., Yang, J. (eds.) *BPM 2008 Workshops.* LNBIP, vol. 17, pp. 649–658. Springer, Heidelberg (2009)
24. Curtis, B., Kellner, M., Over, J.: Process Modeling. *Communication of the ACM* 35(9) (1992)
25. Bruno, G., Dengler, F., Jennings, B., Khalaf, R., Nurcan, S., Prilla, M., Sarini, M., Schmidt, R., Silva, R.: Key challenges for enabling agile BPM with social software. *Journal of Software Maintenance* 23(4), 297–326 (2011)
26. Dengler, F., Lamparter, S., Hefke, M., Abecker, A.: Collaborative Process Development using Semantic MediaWiki. In: *Proceedings of the 5th Conference of Professional Knowledge Management*, Solothurn, Switzerland (2009)
27. Qu, H., Sun, J., Jamjoom, H.T.: Scoop: Automated social recommendation in enterprise process management. *IEEE International Conference on Services Computing* 1, 101–108 (2008)
28. Koschmider, A., Song, M., Reijers, H.A.: Social software for modeling business processes. In: Ardagna, D., Mecella, M., Yang, J. (eds.) *BPM 2008 Workshops.* LNBIP, vol. 17, pp. 666–677. Springer, Heidelberg (2009)
29. Fengel, J., Rebstock, M., Nüttgens, M.: Modell-Tagging zur semantischen Verlinkung heterogener Modelle. In: *EMISA*, pp. 53–58 (2008)
30. Reich, J.: Supporting the execution of knowledge intensive processes by means of expert and best-practice mediation. *Dr. Hut*, München (2008)
31. Brambilla, M., Fraternali, P., Karina, C., Ruiz, V.: Combining social web and BPM for improving enterprise performances: the BPM4People approach to social BPM. In: *WWW (Companion Volume)*, pp. 223–226 (2012)
32. Silva, A.R., Meziani, R., Magalhães, R., Martinho, D., Aguiar, A., Flores, N.: AGILPO: Embedding Social Software Features into Business Process Tools. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) *BPM 2009.* LNBIP, vol. 43, pp. 219–230. Springer, Heidelberg (2010)
33. Bider, I., Johannesson, P., Perjons, E.: A Strategy for Merging Social Software with Business Process Support. In: Muehlen, M.z., Su, J. (eds.) *BPM 2010 Workshops.* LNBIP, vol. 66, pp. 372–383. Springer, Heidelberg (2011)
34. Johannesson, P., Andersson, B., Wohed, P.: Business Process Management with Social Software Systems – A New Paradigm for Work Organisation. In: Ardagna, D., Mecella, M., Yang, J. (eds.) *BPM 2008 Workshops.* LNBIP, vol. 17, pp. 659–665. Springer, Heidelberg (2009)
35. Richter, A., Riemer, K.: Corporate Social Networking Sites – Modes of Use and Appropriation through Co-Evolution. In: *20th Australasian Conference on Information Systems*, Melbourne, paper 34 (2009)
36. Dais, A., Nikolaïdou, M., Anagnostopoulos, D.: A Web 2.0 Citizen Centric Model for T-Government Services. *IEEE Intelligent Systems* (published online July 2012), doi: 10.1109/MIS.2012.63

37. Surowiecki, J.: *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Doubleday (2004)
38. Fleischmann, A., Stary, C.: Whom to talk to? A stakeholder perspective on business process development. *Universal Access in the Information Society* 11(2), 125–150 (2012)
39. Yu, E.S.K., Mylopoulos, J.: From E-R to “A-R” – Modelling Strategic Actor Relationships for Business Process Reengineering. In: Loucopoulos, P. (ed.) *ER 1994*. LNCS, vol. 881, pp. 548–565. Springer, Heidelberg (1994)
40. Granitzer, M., Granitzer, G., Tochtermann, K., Lindstaedt, S., Rath, A., Groiß, W.: Automating Knowledge Transfer and Creation in Knowledge Intensive Business Processes. In: Ardagna, D., Mecella, M., Yang, J. (eds.) *BPM 2008 Workshops*. LNBIP, vol. 17, pp. 678–686. Springer, Heidelberg (2009)
41. Van der Aalst, W.M.P.: *Process Mining - Discovery, Conformance and Enhancement of Business Processes*, pp. I-XVI, 1–352. Springer (2011)
42. Alexopoulou, N., Nikolaidou, M., Kanellis, P., Mantzana, V., Anagnostopoulos, D., Martakos, D.: Infusing agility in business processes through an event-centric approach. *International Journal of Business Information Systems* 6(1), 58–78 (2010)
43. Bider, I., Johannesson, P., Perjons, E.: In Search of the Holy Grail: Integrating Social Software with BPM Experience Report. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) *BPMDS 2010 and EMMSAD 2010*. LNBIP, vol. 50, pp. 1–13. Springer, Heidelberg (2010)
44. Ould, M.: *Business Process Management: A Rigorous Approach*. Meghan Kiffer Pr. (2005)
45. Harrison-Broninski, K.: Modeling Human Interactions: Part 2. BPTrends (2005), <http://www.bptrends.com>
46. Hatzi, O., Nikolaidou, M., Katsivelis, P., Hudhra, V., Anagnostopoulos, D.: Using social network technology to provide e-administration services as collaborative tasks. In: International Conference on Electronic Government and the Information Systems Perspective and International Conference on Electronic Democracy, Vienna, Austria (September 2012)
47. Abowd, G.D., Dey, A.K.: Towards a Better Understanding of Context and Context-Awareness. In: Gellersen, H.-W. (ed.) *HUC 1999*. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999)
48. Ploesser, K., Recker, J., Rosemann, M.: Challenges in the context-aware management of business processes: a multiple case study. In: *ECIS* (2011)
49. Saidani, O., Selmin, N.: Context-Awareness for Adequate Business Process Modelling. In: *IEEE RCIS 2009*, pp. 177–186 (2009)
50. European Interoperability Framework for European Public Services (EIF) ver. 2.0 (2010)
51. Krishnamurthy S.: Social Process Design, Execution and Intelligence for a better Customer Experience. Infosys, white paper (2011)
52. Bessai, K., Nurcan, S.: Actor-Driven Approach for Business Process. How to Take into Account the Work Environment? In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) *BPMDS 2009 and EMMSAD 2009*. LNBIP, vol. 29, pp. 187–196. Springer, Heidelberg (2009)

Context-Aware Agile Business Process Engine: Foundations and Architecture

Irina Rychkova, Manuele Kirsch-Pinheiro, and Bénédicte Le Grand

Centre de Recherche en Informatique, Université Paris 1, Panthéon-Sorbonne
90, rue Tolbiac, 75013, Paris, France
{Irina.Rychkova,Manuele.Kirsch-Pinheiro,
Benedicte.Le-Grand}@univ-paris1.fr

Abstract. Future developments for enterprise process management must evolve from the current systems based on rigid, workflow based processes into context-aware, agile dynamic structures, which exploit local adaptability. In this idea paper, we define two forms of process agility. To enable these forms of agility, we present our vision of context-aware business process management based on declarative modeling combined with innovative context management and formal concept analysis. We finally describe the foundations and introduce the architecture of a context-aware agile business process engine (CAPE).

Keywords: business process agility, context awareness, declarative process modeling, formal concept analysis.

1 Introduction

Capacity to timely discover and to efficiently respond to rapid changes in the environment is a major goal of an *enterprise of the future*. According to [23][4], a firm's ability to adapt to dynamic environments depends first on the agility of its business processes. Therefore, design and development of new process management systems enabling process adaptation at run time are essential.

Lamport defines a process as a sequence of events occurring in system [16], where each event is triggered by an action. Accordingly, a business process can be seen as a sequence of events triggered by activities of business process actors or contextual events. The majority of existing methods for business process design follow *imperative* principles, implying that the order of events is predefined. As a result, all meaningful process events need to be determined and corresponding actions need to be predefined at design time. At run time, processes follow the configured model with limited possibilities to deviate from the predefined scenario.

Execution of a business process in a dynamic environment can be compared to navigating a ship towards its destination bay in uncertain waters. Very rarely can a ship follow blindly a predefined path: awareness about its current position and situation as well as navigation skills and dynamic path finding are essential to reach the destination. This idea paper reports on research, which is currently at its early stage of development. In this work, we discuss foundations and propose architecture for a system supporting dynamic and context-aware business process adaptability.

First, we shift the traditional imperative paradigm for process design and exploit *declarative* principles: we represent a business process as finite state machine (FSM) [22] with a *state* representing a process situation at a given time and *state transitions* defining the possible process scenarios. The *triggering events* specify the underlying process semantics, i.e. conditions for state transitions. The FSM formalism makes the notion of *process activity* implicit while putting forward activity outcomes, which are modeled as triggering events. Therefore, the declarative process model focuses on “what” needs to be done in order to achieve the process goal and not on “how” it has to be done. This allows us to handle process events whose order of occurrence is *undetermined* and to define the corresponding handling scenarios at run time.

Navigation in a stormy ocean depends on a skillful skipper and his capacity to select a right action to ensure that no incorrect scenario is executed. We design initial *navigation rules* for process guidance based on Formal Concept Analysis and Galois lattices [2][6]. We specify the resulting process as a set of activities that can be dynamically assembled at run time into one of the (non-forbidden) process scenarios. In general, such process specification can offer infinitely many alternative scenarios and a possibility to deviate from one scenario to another during the execution. We formalize these properties of a process as the **1st form of agility**.

Navigation in a stormy ocean depends on the capacity of the skipper to select a right action at the right moment, and with respect to the current situation: we define the **2d form of process agility** as the ability to monitor and manage the process context and to dynamically select and/or alter the execution scenario accordingly. We extend the declarative process specifications with dynamic context models and mechanisms for dynamic context management [3][19][20].

We design *navigation rules* for processes guidance that handle both process events (events resulting from execution of process activities) and context events in a unified way. This is compatible with the FSM formalism: the nature of events triggering the state transition has no importance. The navigation rules ensure that no incorrect scenario will be executed with respect to a given context situation.

Novel combination of declarative modeling principles, context-awareness and Formal Concept Analysis is the main research contribution of this work. The architecture for a context-aware business process engine (CAPE) summarizes our findings.

The remainder of this paper is organized as follows: in Section 2, we discuss the state of the art related to business process design and associated agility problems; in Section 3 we formalize two forms of business process adaptability. These theoretical foundations are used in Section 4 to specify the architecture of our context-aware agile business process engine (CAPE). We finally illustrate our findings on an example and present the perspectives of this work.

2 Motivation and Related Work

Workflow-based approaches are highly efficient for process design and management assuming that: (i) all the events triggered by process activities are well determined; (ii) human involvement is limited to “accept”, “reject” or “select from the list” types of decisions; and (iii) the system within which a process is executed can be considered as closed: no external event affecting the process execution can occur.

For a large category of processes, however, these assumptions do not hold: in health care, the same therapy may have different effects on different patients; in insurance, claim processing extensively involves human expertise and decision making; in business, many processes have to cope with evolving economic conditions and frequent changes of customer requirements. The limited capacity of imperative methods to deal with changes has been recognized by both researchers and practitioners [32]. Numerous approaches propose to improve the dynamic process adaptability:

In [26][15][27] the concept of *configurable* process is presented and different modeling formalisms to deal with process configurability are defined. Other works aim at improving run time adaptability through *modification* of the predefined workflow during execution [24]. At run time the process instances follow the preconfigured model with a limited adaptability via predefined variants or deviation.

The Declare framework [21][31] is a constraint-based system that uses a declarative language grounded in temporal logic. This framework supports process flexibility both at design and run time. Compared to our approach, Declare is activity-oriented; contextual information is not considered by this approach.

Solutions for processes characterized by “unpredictability” are reported in numerous works [30][17][1][7]. In [30], the foundations and collection of experience reports on Adaptive Case Management are presented. These works emphasize run time adaptability. Markus et al. [17] propose a framework to support emergent knowledge processes, implemented in the TOP Modeler tool. In [1] process instances are represented as dynamically moving through state space. This approach relies on automated control systems and implements declarative modeling principles. Burkhardt et al. [7] propose to explore the capabilities of recommender systems to provide the user with intelligent process-oriented support at run time. While handling dynamic activity selection and configuration of processes “on the fly”, the majority of proposed solutions demonstrate only limited capacity to deal with process contextual information in systemic and dynamic way.

Soffer and Yehezkel [29] introduce semantics for a declarative process model based on Generic Process Model (GPM). GPM is state-oriented; it captures the process context and reasons about process goals. Though based on different theories, this approach is the most related to the one presented in this paper.

In [25][28], authors use context information for process definition. Roseman et al. [25] consider that external context elements may influence business processes (e.g. weather influences processes of a call center in an insurance company). They incorporate such elements into business process modeling. Saidani et al. [28] also consider context in business process definition, in particular, the roles played by actors. In these works context information is specified only at *design time*. Mounira et al. [18] propose a process mining architecture to identify context variables influencing process activities. However, no specific model formalizing these variables is proposed.

3 Foundations for CAPE

In this section we define two forms of process agility and present our vision of context-aware business process management based on a fully declarative modeling combined with innovative context management and formal concept analysis.

3.1 Process Agility at Work: Agile Patient Care

As we could see in “House” American TV series¹, Patient diagnostics and treatment processes in a medical ward only partially rely on imperative procedures. The main challenge is to be aware of the patient situation and its evolution and to adjust the treatment accordingly. Contextual parameters that might be relevant and should be managed include (but are not limited to):

- Patient’s measurable body conditions (temperature, blood pressure, heart rate);
- Patient’s medical record;
- Patient’s life style;
- Information about recent workload, leisure activities, trips.

Some of these parameters are stable (e.g. predispositions, allergies, etc.), others can evolve (e.g. new information about the patient’s medical history, recent activities), and some others may change several times a day (e.g. body temperature). The capability to immediately react by canceling/prescribing new tests or medications is essential.

3.2 First Form of Process Agility

We define the first form of business process agility as *a capacity to handle unpredictable sequences of system events*. This implies that the order of process activity invocations is defined dynamically, at run time, and depends uniquely on the current situation (process state) rather than on a predefined execution scenario(s).

3.2.1 Declarative Approach to Process Specification

To ensure the first form of agility, we shift the traditional imperative paradigm in process specification and exploit *declarative* principles: we represent a business process as a finite state machine (FSM) – a state-transition system - that allows us to handle process events (and context events – see Section 3.3) with *undetermined* order of occurrence and to define the corresponding scenarios at run time.

A FSM [22] specifies a machine that can be at one state at a time and can perform a state transition as a result of a triggering condition (event or group of events). It is defined by a (finite) set of states and a set of triggering conditions for each transition.

Mathematical model:

A FSM can be defined as a quintuple $\langle Q, \Sigma, \delta, q_0, F \rangle$ where:

¹ http://en.wikipedia.org/wiki/House_TV_series

- $Q = \{S_0, S_1, \dots S_n\}$ - is a finite set of states;
 $\Sigma = \{E_1, E_2, \dots E_m\}$ - is a finite set of triggering events (input alphabet);
 $\delta: Q \times \Sigma \rightarrow P(Q)$ is the state transition relation;
 $S_0 \in Q$ is the initial state;
 $F \subseteq Q$ is the set of final states.

A business process can be modeled as a finite state machine where:

- A FSM *state* $s \in Q$ represents a state of the process at a given time. It is described by the states of its parameters (e.g. order: {in preparation, delivered, paid}, patient: {admitted, in diagnostics, under treatment, discharged}).
- FSM triggering events Σ represent events that may occur during the process execution (e.g. change of patient's body temperature resulting from medication etc.). Note, that process activities are implicit within FSM formalism: only the events resulting from an activity execution are observable.
- FSM transitions δ represent transitions between the process states.
- FSM final states F represent the process outcomes (order delivered, patient discharged, etc.) and can be associated with the process goal.

Representation of a business process using states and transition is not new: [33] presents DecSerFlow language for Declarative modeling of service. Formal semantics for DecSerFlow is based on labeled transition systems. In [1][5], the process execution is presented as a trajectory in the process state space. The process model is specified as a set of formal rules describing the valid paths of the possible trajectories. Our approach extends this paradigm and uses the FSM formalism as a common ground for *process model, context model and formal concept model*.

Example: Agile Patient Care (continued)

To illustrate the FSM formalism, we develop our example on agile patient care and model a (simplified) patient treatment process as a FSM illustrated in Fig. 1:

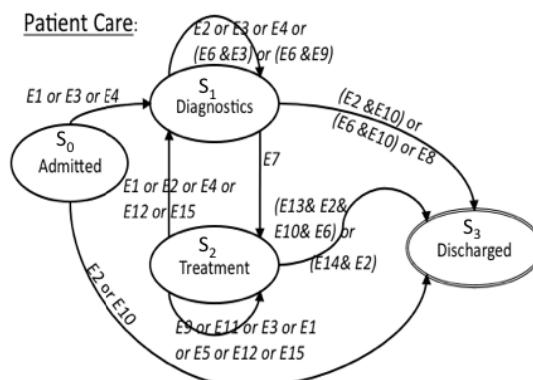


Fig. 1. A finite state machine (FSM) representing a patient treatment process. The events E and the state transitions they trigger are shown as labels

After a patient is *admitted* (S_0) to a medical ward (the initial state), a physician examines him in order to obtain information for *diagnostics* (S_1) and further *treatment* (S_2). Diagnostics may involve one or multiple examinations and/or generic or specific tests. The patient's case is then assessed and a *treatment* is prescribed. During the treatment, additional examinations can reveal new patient's condition and require to modify the assigned therapy and, even, to repeat diagnostics and assessment. Once the therapy is terminated and the patient's good condition is confirmed, the patient is *discharged* (S_3) from the ward. In this example, we identify four states: $Q=\{S_0, S_1, S_2, S_3\}$ and six process activities that can be executed during the process and trigger state transitions from S_0 :*Admitted* to S_3 :*Discharged* states (Table 1).

Table 1. Abstract activities and events defined for the Patient treatment process

Activity	Avail. at:	Process events (Activity outcomes):	
A1 Physical examination	S_0, S_1, S_2	E1 Confirms the declared symptoms E2 No problem found E3 New symptoms emerged E4 Supplementary medical tests are required	
A2 Medical laboratory test	S_1, S_2	E5 Positive results (anomalies detected) E6 Negative results (no anomalies detected) E4 Supplementary medical tests are required	
A3 Specific medical tests	S_1, S_2	E5 Positive results (anomaly detected) E6 Negative results (no anomaly detected) E4 Supplementary medical tests are required	
A4 Case Assessment	S_1	E7 Diagnose confirmed, treatment assigned E8 Diagnose not confirmed, patient discharged E4 Supplementary medical tests are required	
A5 Therapy	S_2	E9 Condition declined (e.g. symptoms increasing) E10 Condition improved (e.g. symptoms decreasing) E11 Side effects emerged E3 New symptoms emerged E12 Stable situation E13 End of therapy	
A6 Recovery	S_2	E3 New symptoms emerged E14 End of recovery therapy	
Context events:			
		E15	New medical / personal evidence received
		E3	New symptoms emerged
		E9	Condition declined, (e.g. symptoms increasing)
		E10	Condition improved (e.g. symptoms decreasing)

According to our formalism, an activity A is described with a pair $\langle S, E_A \rangle$ where:

- $S \subseteq Q$ is the set of states from which this activity can (but not necessarily will) be invoked;
- $E \subseteq \Sigma$ is the set of events that can result from the activity execution and can potentially trigger a state transition.

For each activity A, the state transitions that can be triggered upon its termination can be calculated as: $\delta_A: S \times E \rightarrow P(S)$.

For example, the activity A2 (Medical laboratory test) is specified as follows: S: {S1, S2}; E:{E5, E6, E7}. Note that some events can result from a process activity and can be context events (cf. Section 3.3) - independent from activities. (e.g. E10 – condition improved).

3.2.2 Formal Concept Analysis and Galois Lattices

Within our model, the partial ordering of process activities is determined by the state transition relation P(Q). This relation specifies the valid transitions with respect to the process goal (its final state) and ensures that invalid state transitions (e.g. to *discharge* a patient with *critical* temperature) will be avoided. This relation can be specified with *Formal Concept Analysis* (FCA) [2] [6]. FCA is a mathematical theory relying on the use of formal contexts² and Galois lattices defined below. The use of Galois lattices to describe a relation between two sets has led to various classification methods [8] [35]. Since then, they have been used in numerous areas to extract hidden knowledge from data. Let us first introduce FCA terminology [12].

Mathematical model:

Let $K = (G, M, I)$ a formal context, where G is a set of *objects*, M is a set of *attributes*, and the *binary relation* $I \subseteq G \times M$ specifies the attributes of the different objects. Derivation operators noted $(.)^l$ are defined for $A \subseteq G$ and $B \subseteq M$ as follows:

$$A^l = \{m \in M \mid \forall g \in A : g \text{ } Im\} \quad B^l = \{g \in G \mid \forall m \in B : g \text{ } Im\},$$

where A^l is the set of attributes, which are common to all objects from A and B^l is the set of objects which share all attributes from B .

A *formal concept* of the formal context (G, M, I) is a pair (A, B) , where $A \subseteq G$ and $B \subseteq M$, $A = B^l$ et $B = A^l$. The set A is called the *extent* of concept (A, B) and B is its *intent*. A concept (A, B) is a specialization of concept (C, D) if $A \subseteq C$, which is equivalent to $D \subseteq B$. This is noted $(A, B) \leq (C, D)$. Reciprocally, the concept (C, D) is a generalization of concept (A, B) . The set of all concepts and their partial order relation constitutes a lattice, called Galois lattice of the formal context K .

The major interest of a Galois lattice is the structure it provides through the conceptual clustering of objects according to their common attributes. This allows the identification of the most conceptually significant objects and attributes. Another interest of Galois lattices is that association rules can be inferred automatically from them. Several works have indeed applied FCA to the extraction of relevant association rules [13] or to perform sequential pattern mining [11].

Within our approach, process states Q , triggering events Σ and process activities defined in Section 3.2 form a formal context and can be analyzed using Galois lattices. Process states and activities can be clustered revealing their conceptual properties: For example, we can determine activities that can be executed (or suggested for execution) under given conditions and with an objective to trigger a desired state transition.

² The term “formal context” is specific to Formal Concept Analysis and refers to a binary relation. In the following, we will also refer to this as “formal context”, as opposed to “context” which represents the environment.

Fig. 2 represents Galois lattices built from the formal context defined in Table 1. The lattice in Fig 2a clusters activities into (possibly overlapping) groups according to the common events shared by the activities within each group. Each node in the line diagram represents a concept of the lattice; its label is a couple of sets corresponding respectively to the extent and the intent of the concept. For example, the concept labeled as $\{(A_2, A_3); (E_4, E_5, E_6)\}$ contains the activities A_2 and A_3 which may all lead to events E_4 , E_5 and E_6 . The lattice in Fig 2b represents the sets of activities enabled for each state³.

We exploit Galois lattices as a recommender mechanism, which makes suggestions about activities to execute. This mechanism is explained in Section 4. Each concept of the lattice contains a set of activities (white labels) described by the events they share (grey labels). The links between concepts are generalization/specialization links. Note that nodes inherit events from concepts above them and that they inherit activities from concepts below them. The lattice has an upper bound (top concept), which contains all activities (through inheritance) but no event (i.e. there is no event associated with all activities). The lower bound of the lattice (bottom concept) contains no activity and all events.

Let us consider the concept containing only activity A_1 and the concept containing only A_4 in its extent (Fig. 2a). We see from the intent that A_1 may have events E_1 , E_2 , E_3 or E_4 as its outcome and A_4 may lead to events E_4 , E_7 or E_8 . These two concepts are generalized by the concept $\{(A_1, A_2, A_3, A_4), \{E_4\}\}$, which contains activities A_1 , A_2 , A_3 and A_4 , all described by the common event E_4 . Along those lines, the concept $\{(A_1, A_2, A_3), \{S_1, S_2\}\}$ in Fig.2b shows the activities described by common states S_1 and S_2 – the states where these activities can be executed.

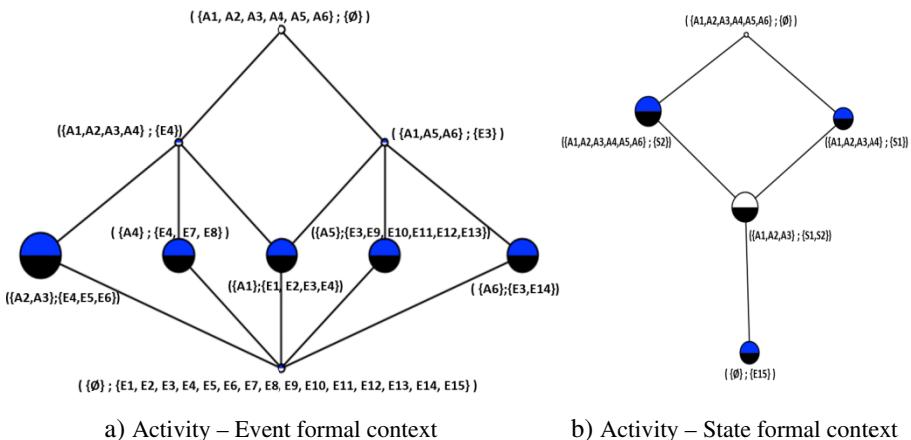


Fig. 2. Galois lattices

³ We have separated events from states by creating 2 distinct lattices to make our methodology clear, as states are preconditions for activities whereas events are possible outcomes of these activities. It should be noted however that events and states may be associated to activities within a single formal context and therefore a unique Galois lattice.

3.3 Second Form of Process Agility

We define the second form of business process agility as *a capacity to adjust the process execution scenario according to the current contextual situation*. Process activities are assembled at run time, according to observed context⁴ and with an objective to trigger a state transition required for achieving the process goal (defined as one of the final states of a FSM).

Dey [9] defines a context as *any information that can be used to characterize the situation of an entity (a person, place or object) that is considered relevant to the interaction between a user and an application*. The notion of context adopted in the literature is mostly user-centric and limited to physical aspects (e.g. location, user preferences, or user device)[19]. Together with Dourish [10], we argue that the notion of context is larger, and includes information related to organization and processes: “context – the organizational and the cultural context, as much as the physical context – plays a critical role in shaping action and also in providing people with the means to interpret and understand action”. In our example, patient treatment process can be influenced by the emergence of new symptoms or the arrival of new resources (e.g. new medical people available, new personal evidence, etc.). The second form of business process agility consists in taking into account such context information during process execution.

The context parameters reflect our awareness about external and internal information about the process; they can be observed and measured. Even though context-awareness for business processes is addressed in the literature [25] [28] [18], the lack of formalism for context representation and management persists: many of the proposed context models are static (need to be defined at design), incomplete (consider only limited context information) and are often specific to workflow-based processes.

We argue that the number and kind of context parameters may vary from one situation (or process state) to another making it impossible to exhaustively model all required context information within a single (static) context model. The context model, therefore, needs to be dynamically instantiated from an appropriate metamodel according to specific (evolving) context dimensions.

3.3.1 Dynamic Context Modeling

The way context information can be exploited for business process flexibility depends on what information is observed and how it is represented. According to Najar et al. [19], the formalism chosen for representing context model determines the reasoning methods that can be used to perform system adaptation to the observed context. A context model (i) ensures the definition of independent adaptation processes and (ii) isolates this process from context acquiring techniques. The same applies to context-aware business process. We claim that the process context information should be acquired, modeled and formally analyzed at run time in order to adapt business process execution and to ensure business process flexibility.

⁴ The term of “context” represents the environment.

Several context models have been proposed in the literature [19] [3]. Based on these, we define a common meta-model presented in Fig. 3. In this meta-model, we consider context as a set of context elements that are observed for a given subject (e.g. the patient, a device, a resource, etc.). Each subject can be associated with multiple context elements (location, status, etc.); for each context element, we observe values that can dynamically change. Subject and context elements can be semantically described using ontologies. Ontologies are considered as interesting to represent context elements [19]. They provide ways to semantically describe these elements and their relationships, but also reasoning mechanisms, notably inference rules.

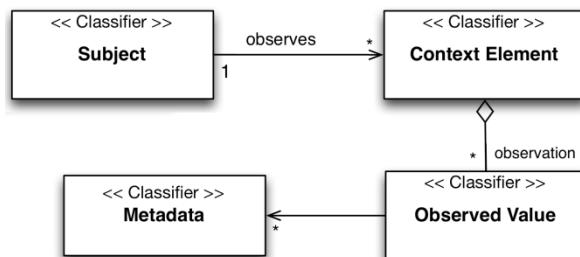


Fig. 3. Context meta-model considering context as a set of context elements

Mathematical model:

Based on context ontology, we represent events as logical constraints over context concepts and observed values. We formalize the context of a subject s in a time t as follows:

$$\text{Context}(s,t) = \{ \text{Element}(s, ce) \},$$

where $\text{Element}(s,ce)$ represents the observed value of the context element ce for the subject s .

Within our approach, process states Q defined using FSM formalism in Section 3.2 can be extended with the context information: Each state can be associated with a (set of) subject and its contextual elements to observe. FSM triggering events Σ represent both (i) *context events* occurring during the process execution and/or (ii) events resulting from executed process activities – *process events*. Note that process activities are implicit within FSM formalism: only the events resulting from an activity execution are observable. As a result, a FSM processes both *contextual* and *process* events in a unified way. These events can be expressed using logical conditions on observed values of contextual elements.

In our example, a subject s – patient – is observed in all the process states. The contextual elements ce associated with a patient include his body temperature, blood pressure, patient's location, record etc. Patient's condition can decline (patient's temperature may evolve from ' 36.5°C ' to ' 39.7°C ') or new evidence about the patient can be received by e-mail. Triggering event E9 (condition declined – see Table 1) can be expressed using the following condition: $\text{Element}(\#patient, \#temperature) > 38.5$.

3.3.2 Context Model and Concept Analysis

Fig. 1 shows both the process and the context events that are taken into account by the patient treatment process. Context events affect state transitions and play the role of triggers, similarly to activity outcomes. For example, the patient treatment (S2) can be changed either as a result of physical examination outcomes (E1) or as a response to the new symptoms (E11), or because a new evidence about the patient (e.g. allergies, predispositions, past incidents, etc.) (E15).

Galois lattices are particularly adapted to our context-aware approach: we use Galois lattices to analyze FSM triggering events (see Section 3.2) regardless their nature (process events or context events). Indeed, FCA has already been successfully coupled to context modeling [34]. The authors have proposed a spontaneous, personalized and dynamic way of defining and joining user communities. They use a lattice-based classification and recommendation mechanism that analyzes the interrelations between communities and users, and recommends new communities, based on user interests and preferences.

4 Architecture for CAPE

The functionality of a process engine assuring the agility forms introduced above can be described as follows: this engine enables the activities for execution, captures the events (internal or external) and generates state transitions (Fig. 4). Considering that the process objective is specified by its final state (or states), the engine should also assure an appropriate guidance for the decision maker (a human agent): it recommends him the activities that would maximize chances to trigger a transition leading towards the process goal.

We define the following primary elements for a context-aware process engine: activity repository, context monitor and navigation manager, as illustrated in Fig. 4.

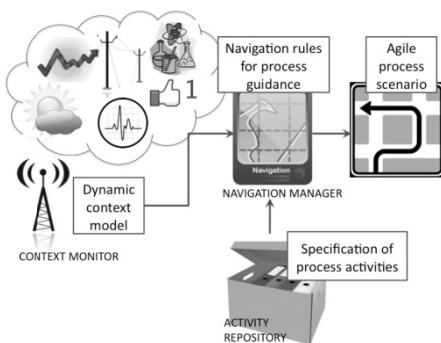


Fig. 4. CAPE architecture: the context monitor, the activity repository and the navigation manager

4.1 Activity Repository

Activity repository represents a set of activities related to a business process. Each activity is specified with states S from which it can (but not necessarily must) be executed during the process, and with the set of events E representing its possible outcomes as defined in Section 3.2.1. The pair $\langle S, E \rangle$ is an activity signature. These signatures are further used by Navigation Manager. The activity repository represents crew skills and technical capacity of a ship.

4.2 Context Monitor

Context monitor is in charge of observing, at run time, context elements and subjects from the environment. Its role in CAPE architecture is similar to a watchman in a ship: it observes navigation conditions and reports them to the navigation manager.

Context monitor is based on the context meta-model described earlier. Similar to [20], it recognizes context elements through plugins (software components dynamically connected to the architecture), which feed the architecture with dynamic information about a given context element from a subject (e.g. a plugin reading patient's heart rate or medical resource's location from his/its id card). Context monitor can be dynamically extended, by adding new plugins, for observing new context elements and subjects. Context values dynamically observed by context plugins measure the process position in the process state space (for instance, a new plugin for observing the availability of a new analysis equipment). This position is further used by the navigation manager, which recognizes context events and interprets them accordingly.

4.3 Navigation Manager

Navigation manager makes navigation decisions, like a skipper in a ship. It determines one (or several) plausible activity to execute with respect to the process goal and the contextual situation. Specifically, navigation manager takes into account the current state of the process, the contextual parameters and the signatures of activities defined in the activity repository. Based on the *navigation rules*, it determines a set of activities *enabled* in a given situation and calculates those of them that will have a highest probability to result in the desired outcome.

Navigation engine is the core element of CAPE architecture: it links together the other elements. We illustrate the functioning of CAPE on our example:

Example: Agile Patient Care (continued)

Once a patient has been admitted (S_0) and the result of his physical examination (A_1) has confirmed the declared symptoms (E_1), he is in state S_1 . The targeted final state is S_3 : discharge the patient. The goal of the navigator is to provide recommendations in order to reach this state as quickly as possible, while respecting medical protocols and taking into account the contextual situation. In the following, we describe the sequence of operations performed by the navigator to meet this goal.

1. Selection of the next transition towards the targeted final state

Fig. 1 illustrates the various possible paths from one state to another. The underlying graph is directed, nondeterministic, with possible cycles. From this graph the

navigator computes the most efficient path from S1 to S3, both in terms of length and of events triggering probability. Let us suppose here that this ideal path consists in using the direct link from S1 to S3. The next step, in our example, is to trigger the transition from S1 to S3.

2. Identification of relevant activities

The navigator now has to identify the events, which may⁵ trigger a transition from S1 to S3. According to our FSM (Fig.1) and the state transition map, the triggering conditions for a transition S1 - S3 are the following:

- Additional physical examinations and/or medical laboratory tests reveal no anomaly or problem and the patient himself feels better - (E2 | E6)&E10, or
- Case assessment does not confirm the diagnosis (E8)

To identify the activities that can be of a maximum utility with respect to our objective (transition S1-S3):

- Navigator identifies the activities in our repository that can ensure a desired outcome (a combination of events described above) using the Galois lattice illustrated in Fig. 2. For our example, the event E2 can result from the activity A1; E6 from A2 or A3; E10 from A5 or spontaneously as an external event, as the patient's condition may improve independently from any activity from the medical staff.
- Navigator selects those activities that are enabled at the state S1 (according to their specification) using the Galois lattice illustrated in Fig. 2. For our example, the activities A1, A2, A3, A4 are available in S1 (In Diagnostic) state, whereas A5 is not.

Thus, the activities A1 – A4 are potential candidates for execution in the state S1.

3. Selection of recommended activities based on their utility

We calculate an utility of each scenario as its likelihood to trigger the transition S1-S3. For our example, A5 is not available at S1 and therefore, E10 can be expected only as a context event (i.e. we can observe this event when it happens but cannot control it). The following viable scenarios can be evaluated:

- a) A physician prescribes medical tests and/or makes additional physical examinations according to the declared symptoms. The spontaneous improvement of the patient is expected.
- b) A physician assesses the case based on the patient's history (and examinations made upon patient's admission to the ward).

A combination of these scenarios is also possible.

Depending on the probabilities of transition S1-S3 resulting from each of the aforementioned scenarios and also, on the probability to obtain the event E2 as a result of A1, E6 from A2 or from A3, and E8 from A4, the navigator may recommend specific activities to the medical staff.

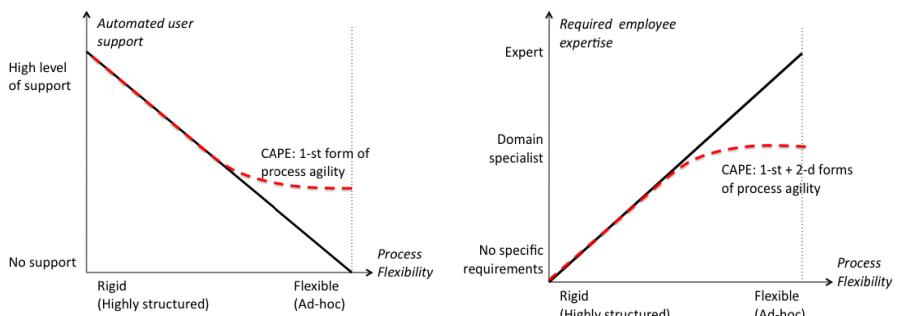
⁵ Our graph is nondeterministic: one event can trigger multiple state transitions (e.g. E11 – side effects emerged – can be handled both in the therapy states or can trigger a transition to the diagnostics state. This nondeterminism, in general case, can be resolved by refining the states and transitions, and by specifying the new navigation rules. Alternatively, a probability p can be associated with each of the possible transitions.

5 Discussion and Future Work

While providing a rich toolbox for process modeling, verification, and post-execution analysis, traditional workflow-based information systems are considered to be inflexible. Therefore, organizations nowadays are searching for a compromise between the automated user support and process flexibility at run time. Fig. 5a illustrates this dependency: the x-axis shows a degree of process flexibility (from highly structured to unstructured, ad-hoc processes); the y-axis shows a level of automated user support that can be assured by the system. The solid line depicts the dependency: the more flexible a process, the lower the user support provided for it; and conversely: the higher the expected level of user support, the more structured a process has to be.

According to Burkhart et al. [7], “With increasing runtime flexibility, employees are confronted with an expanding decision space and they are needed to possess more expertise in dealing with the processes they are involved in.” Fig. 5b illustrates this dependency: the x-axis shows a degree of process flexibility; the y-axis shows a level of user expertise required for handling this process. The solid line depicts that more flexible a process is, the higher expertise level of an employee should be.

The two forms of agility formulated in this paper aim at relaxing the requirements both for human expertise for a given level of process flexibility and for process rigidity (predefined structure) for a desired level of automation. *Declarative approach* for process design and use of formal methods enable a set of automated techniques for process analysis and validation based on model checking and theorem proving. Thus, they improve the level of automated user support allowing maximum run time flexibility. *Context awareness and formal concept analysis* enable automated recommendations and identification of alternatives. Their joint use provides flexible guidance to end users at run time and supports them with an expertise required for the process handling.



- a) Process flexibility vs. Automated user support
- b) Process flexibility vs. Level of employee expertise required

Fig. 5. Relations between process flexibility and automated support / required level of user expertise. The solid line depicts the current trends in BPM; the dashed line depicts how CAPE architecture presented in this work can possibly change these trends

This idea paper reported on research, which is currently at its early stage of development. According to the IS research framework [14], (i) we specified our **business problem** as a lack of automated support for business process agility; (ii) then we defined a relevant **knowledge base** for our research and outlined the foundations for CAPE; (iii) we **built our design artifacts**: we introduced the two forms of process agility – **the constructs** to be used for reasoning about business processes; we also defined **a model** and **a method** for specification of agile business processes based on FSM abstraction and formal concept analysis. We extended this model with the dynamic context model. We combined these artifacts in and proposed the architecture for context-aware agile business process engine (CAPE).

This work accomplishes the first part of the “build-evaluate” loop [14]. **Evaluation** of our designed artifacts and their **refinement** will be addressed in future work. More specifically, we envisage to demonstrate the utility of our proposed architecture, first, by developing detailed scenarios, then, by simulating them, and, eventually, by implementing CAPE architecture and studying its usability in real business environment. Usability metrics for CAPE will be also discussed in future publications.

References

1. Andersson, T., Bider, I., Svensson, R.: Aligning people to business processes experience report. Software Process: Improvement and Practice (2005)
2. Barbut, M., Monjardet, B.: Ordre et classification. Algebre et combinatoire, Tome 2, Hachette (1970)
3. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing 6(2), 161–180 (2010)
4. Bider, I., Johannesson, P., Perjons, E.: Do workflow-based systems satisfy the demands of the agile enterprise of the future? In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBP, vol. 132, pp. 59–64. Springer, Heidelberg (2013)
5. Bider, I.: Towards a Non-workflow Theory of Business Processes. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBP, vol. 132, pp. 1–2. Springer, Heidelberg (2013)
6. Birkhoff, G.: Lattice Theory, 1st edn. Amer. Math. Soc. Pub., Providence (1940)
7. Burkhardt, T., Weis, B., Werth, D., Loos, P.: Towards Process-Oriented Recommender Capabilities in Flexible Process Environments—State of the Art. In: 45th Hawaii Int. Conf. on System Science, HICSS, pp. 4386–4395 (2012)
8. Carpineto, C., Romano, G.: Galois: An order-theoretic approach to conceptual clustering. In: 10th Conf. on Machine Learning, pp. 33–40. Kaufmann (1993)
9. Dey, A.: Understanding and Using Context. Personal and Ubiquitous Computing 5, 4–7 (2001)
10. Dourish, P.: Seeking a foundation for context-aware computing. Human Computer Interaction 16(2-4), 229–241 (2001)
11. Egho, E., Jay, N., Raissi, C., Napoli, A.: A FCA-based analysis of sequential care trajectories. In: 8th Int. Conf. on Concept Lattices and their Applications (2011)
12. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
13. Hamrouni, T., Ben Yahia, S., Nguifo, E.M.: GARM: Generalized Association Rule Mining. In: CLA 2008, pp. 145–156 (2008)
14. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly 28(1), 75–105 (2004)

15. La Rosa, M., Dumas, M., ter Hofstede, A.H.M., Mendling, J.: Configurable multi-perspective business process models. *J. Information Systems* 36(2) (2011)
16. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM* 21(7), 558–565 (1978)
17. Markus, M.L., Majchrzak, A., Gasser, L.: A design theory for systems that support emergent knowledge processes. *Mis Quarterly* 26, 179–212 (2002)
18. Mounira, Z., Mahmoud, B.: Context-aware process mining framework for Business Process flexibility. In: iiWAS 2010, Paris (2010)
19. Najar, S., Saidani, O., Kirsch-Pinheiro, M., Souveyet, C., Nurcan, S.: Semantic representation of context models: a framework for analyzing and understanding. In: 1st Workshop on Context, Information and Ontologies (CIAO 2009), European Semantic Web Conference (ESWC 2009), pp. 1–10 (2009)
20. Paspallis, N.: Middleware-based development of context-aware applications with reusable components, PhD Thesis, University of Cyprus (2009)
21. Pesic, M., Schonenberg, H., van der Aalst, W.M.: DECLARE: Full support for loosely-structured processes. In: 11th IEEE Int. Enterprise Distributed Object Computing Conference, EDOC 2007, pp. 287–287 (2007)
22. Plotkin, G.D.: A structural approach to operational semantics (1981)
23. Raschke, R.L., David, J.S.: Business process agility (2005)
24. Reichert, M., Rinderle, S., Dadam, P.: ADEPT workflow management system: In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 370–379. Springer, Heidelberg (2003)
25. Rosemann, M., Recker, J., Flender, C.: Contextualization of Business Processes. *Int. J. Business Process Integration and Management* 1(1), 46–60 (2007)
26. Rosemann, M., van der Aalst, W.M.P.: A Configurable Reference Modelling Language. *Information Systems* (2007)
27. Rychkova, I., Nurcan, S.: Towards Adaptability and Control for Knowledge-Intensive Business Processes: Declarative Configurable Process Specifications. In: 44th Hawaii Int. Conf. on System Sciences, HICSS, pp. 1–10 (2011)
28. Saidani, O., Nurcan, S.: Towards Context Aware Business Process Modeling. In: 8th Workshop on Business Process Modeling, Development, and Support (BPMDS 2007), CAiSE 2007 (2007)
29. Soffer, P., Yehezkel, T.: A state-based context-aware declarative process model. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 148–162. Springer, Heidelberg (2011)
30. Swenson, K.D.: Mastering The Unpredictable: How Adaptive Case Management Will Revolutionize The Way That Knowledge Workers Get Things Done, p. 354 (2010)
31. Van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development* 23(2), 99–113 (2009)
32. Van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. *Data & Knowledge Engineering* (2005)
33. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
34. Vanrompay, Y., Ben Mustapha, N., Aufaure, M.A.: Ontology-based User Preferences and Social Search for Spoken Dialogue Systems. In: 7th Int. WS. on Semantic and Social Media Adaptation and Personalization, pp. 113–118 (2012)
35. Wille, R.: Line diagrams of hierarchical concept systems. *Int. Classif.* 11 (1984)

Business Process Workarounds: What Can and Cannot Be Detected by Process Mining

Nesi Outmazgin and Pnina Soffer

University of Haifa, Carmel Mountain 31905, Haifa, Israel
Nesi@Zefat.ac.il, spnina@is.haifa.ac.il

Abstract. Business process workarounds are specific forms of incompliant behavior, where employees intentionally decide to deviate from the required procedures although they are aware of them. Detecting and understanding the workarounds performed can guide organizations in redesigning and improving their processes and support systems. Existing process mining techniques for compliance checking and diagnosis of incompliant behavior rely on the available information in event logs and emphasize technological capabilities for analyzing this information. It is therefore not certain that all the forms of workaround behavior are addressed. In contrast, the paper builds on a list of generic types of workarounds found in practice, and explores whether and how they can be detected by process mining techniques. Results obtained for four workaround types in five real-life processes are reported. The remaining two types are not reflected in events logs and cannot be detected by process mining.

Keywords: Business process workarounds, Process mining, Compliance checking.

1 Introduction

Business processes are automated and managed by organizations in order to streamline and standardize their operations in an effective and efficient manner. However, the standard prescribed procedures are not always followed by employees, and are many times bypassed and worked around.

Addressing such situations is related to the general area of compliance management, which has drawn much attention in recent years [5]. In the general area of compliance management, several types of activities have been identified [11][12][20]. In particular, compliance checking, which checks whether certain constraints are or will be met, and compliance improvement. Compliance checking can be further divided to forward compliance checking, targeting the design and implementation of processes where compliance is enforced, and backward compliance checking, focused on the detection and diagnosis of non-compliant behavior. Compliance improvement modifies the process to improve compliance. This can be done based on diagnostic information resulting from backward compliance checking, and with the use of forward compliance checking techniques.

This paper focusses on backward compliance checking. Yet, as opposed to the general area of compliance management, which refers both to internal policies and external regulations, the focus of this paper is on situations where employees are aware of the required internal procedures and intentionally decide to act differently. We term these situations business process workarounds. As an example, consider a situation where a customer is urgently requesting some goods and a truck is about to embark in his direction. An employee might decide to immediately load the goods on the truck, while the "paperwork" of registering the order and the delivery will be done afterwards in retrospect.

Workarounds are generally considered as a negative phenomenon, assuming the standard process has been designed and optimized to achieve desired business performance. However, since these are intentional actions of employees, we assume they are performed for certain reasons. According to [16], workarounds can be motivated when the defined business processes are rigid and not designed to accommodate situations that might arise, requiring an appropriate response. Additionally, workarounds might be performed when the process design or its support system do not satisfy all the stakeholder needs and expectations. Additional cases might be when employees decide to act upon their own personal goals rather than to follow the defined procedures.

Detecting workarounds and investigating the reasons that drive them can serve organizations striving to compliance improvement and to the design of better processes where workarounds will be reduced. Corrective actions can include process redesign, focused improvement of the business process support system, focused training of the employees, or disciplinary actions.

Various compliance checking techniques have been proposed in recent years as part of the process mining stream of research [1]. These techniques utilize event logs for detecting incompliance to specific constraints, procedures, and process models [3][6][21]. As discussed above, workarounds, as specific forms of incompliance, can be detected using these techniques.

However, the starting point of these techniques is the event log and the technology capabilities. It is therefore not certain that all the forms of workaround behavior are addressed. In contrast, this paper takes a list of six generic workaround types, which were found to exist in business processes [16] as a starting point. Our aim is to explore whether and how workarounds of each of these types can systematically be revealed based on an event log. Building on generic workaround types captures the intentional aspect of workarounds and enables distinguishing them from other types of incompliance. Moreover, it enables us to look for specific patterns that may exist in the log, and to understand what types of workarounds cannot be detected based on the log, if any.

Note that our goal is not to develop new mining techniques. Rather, we wish to explore the capabilities of current technology, commercially available to organizations facing the given workaround types. To this end, we have used Fluxicon Discovery platform (<http://fluxicon.com/disco>) and applied it to logs of five processes taken from three organizations over two years. To generalize the findings, we further discuss capabilities of state-of-the-art technology for addressing these situations.

The remainder of the paper is structured as follows. Section 2 presents the six generic workaround types identified by [16]. Section 3 discusses the patterns that

should be detected in logs with respect to each of the workaround types; Section 4 reports the findings that were obtained for five real-life processes and discusses them. Related work and available state-of-the-art technologies are discussed in Section 5, and conclusions are given in Section 6.

2 Generic Workaround Types

This section presents the six generic workaround types that were identified by [16] in a qualitative study performed in several organizations.

Type A – Bypass of Process Parts

In these workarounds, parts of the process are bypassed, so activities that should be performed at later steps of the process are performed before their time. The activities that were bypassed can be performed in retrospect, or skipped altogether.

As an example, consider a purchasing process, where a participant places a purchase order, and only afterwards initiates the formal approval process.

According to [16], this workaround type appears to be common in practice, and is associated with many situational factors that may indicate reasons that drive its performance. Some factors are related to the process support system, e.g., poor user friendliness and a lack of integration among systems. Other factors relate to process design, which can be complicated and cumbersome, hard to understand, involving many different roles, or not in line with the actual needs and the way the process is practiced. Poor information flow and a lack of feedback about the process status to the process initiator, combined with delays and long execution times, are major drivers that motivate employees to commit workarounds of this type.

Type B – Selecting an Entity Instance That Fits a Preferable Path

This type of workaround relates to situations where a "legitimate" process execution is performed, but the entity instance that is used does not represent the actual one. Rather, it is chosen in order to comply with the transition conditions of the process. As an example, consider a purchase approval process, where transition conditions require additional approvals if the price is over a certain threshold. Employees who know the rules might split purchase requests, whose price exceeds the threshold, and place several requests, each at a relatively small price, to avoid long approval trails.

Usually, the process participants who perform this type of workaround are experienced and knowledgeable, thus they are familiar with the "rules of the game". Consequently, the workarounds are performed systematically and sophisticatedly. These workarounds are mainly associated with complicated and inflexible transition conditions defined in the process.

Type C – Post Factum Information Changes

This type refers to situations where process participants modify data values after these have been used for decision making. There are two variants of this workaround type. First, the data modifications reflect values which were known a-priori and falsely entered with the intention of manipulating the decision making.

For example, in a purchase requisition approval process participants give false information (amounts, purchase items, suppliers, quantities, etc.) which allows the process to move "smoothly" and quickly, and only once the approval steps are completed do they change the information to reflect the real needs. Entering the correct information at the initial stage would have required a different path of approvals and control. Similarly to workarounds of the previous workaround type (B), these workarounds are performed sophisticatedly by experienced employees, who exploit loosely defined access control policies and poor authorization management.

The second (less severe) variant of this workaround type is when the modifications reflect new information or error correction, but no re-iteration of the previous decision is made. This can stem from low awareness of the implications of deviating from the required procedures, as well as poor control policies.

Type D – Incompliance to Role Definition

In this type of workaround, participants perform operations which are not under their responsibility. As an example, consider again a purchasing process, where the initiating participant opens a purchase requisition. When the requisition is approved, it should be handled by a purchasing clerk, who obtains price quotations and selects a winning supplier. A workaround would be when the initiating participant makes inquiries and selects a supplier, and only then transfers the requisition to the purchasing department with the results ready for continued handling.

According to [16], these workarounds typically occur when responsibility assignment does not match (or is not conceived as matching) the knowledge required for certain tasks (e.g., a purchase clerk might not have sufficient technical knowledge to evaluate the available product configurations). Additionally, it might stem from a lack of clear responsibility definitions at different parts of the process. One possible consequence is a poor level of control (incompliance to the "four eyes rule").

Type E – Fictitious Entity Instances

Workarounds of this type are usually performed by employees to compensate for missing or incomplete process definition and support. When certain process steps or variants exist but are not managed and supported within the scope of the process, to gain the possibility of monitoring and documentation, fictitious entity instances are created. These instances are marked (e.g., ItemID 99999) and serve the employees for keeping trace of the unsupported parts of the process.

As an example, in a student intake process, it is impossible to perform an acceptance interview with a candidate before he registers (and has a record). However, the candidate might not wish to register before an interview takes place. To overcome this, the secretary creates a fictitious registration in order to continue the process and invite the candidate for an interview. She immediately assigns the candidate to a fictitious room (to mark that the candidate is awaiting an interview).

Although the intention that drives workarounds of this type is to improve the performance of the process, overcoming problems and increasing the level of control, it is still an intentional (and systematic) deviation from the defined procedures.

Type F – Separation of the Actual Process from the Reported One

In this workaround type, at a certain stage the process participants continue the process manually, possibly until the process is completed. At a separate point in time, the actions that were performed (or should have been performed) are reported in an orderly manner. This is done in a post-hoc manner, only for the purpose of documentation and reporting.

An example of this kind of workaround can, again, be found in a purchasing process. Assume a purchase requisition is waiting for a manager's approval. This might take some time, although the chance that the requisition will not be approved is extremely low to non-existent. Facing this, process participants might not wait for the desired approval and rather move forward with the actual process. Once the approval is obtained, the actions that have been performed (e.g., ordering from the supplier) can be recorded in a post-hoc manner.

Situations where such workarounds are performed are characterized by a high number of administrative steps that do not make real contribution or affect the achievement of the process goal, especially if these steps are likely to cause delays and entail long waiting times. It also appears that workarounds of this type are common when the process moves back and forth between organizational units.

3 Detecting Workarounds in an Event Log

This section examines whether the workaround types discussed above can be detected in an event log, and how. We discuss each workaround type and when possible, specify conditions that should indicate its occurrence in an event log.

Type A – Bypassing Process Parts:

This type is characterized by skipping and bypassing certain process parts. Process instances where such workarounds take place are, hence, incompliant with the prescribed process model, and can be identified using compliance checking techniques. Yet, not every incompliant behavior can be classified as workaround of this type. Specifically, we are looking for activities that are performed while their immediate predecessor (or predecessors) required by the process model have not been performed. The immediate predecessors of an activity can be another activity (if it is in a sequence), several alternative activities (in case the activity follows an OR merge), or several activities that should all be performed (in case the activity follows a synchronization point). We denote the collection of these as $\text{PR}(a)$ – the set of immediate predecessors of activity a .

Consider a trace where activity a appears in the i th position. If for all $r \in \text{PR}(a)$, r is not included in positions $1..i-1$ of the trace, then the trace includes a type A workaround (bypassing process parts). This is checked for all the activities in the trace.

Note that this is a general condition, and it might be too coarse-grained to capture all bypass cases. However, it can be refined and tailored for specific situations. Specifically, it might be required to check the existence of the immediate predecessors of an activity only in part of the trace, after a certain point. For example, if the process includes loops, the immediate predecessors should be found in the trace between consecutive occurrences of the activity.

Fig. 1 provides an example of a mined model of a purchase requisition approval process, where bypasses are marked (arrows 1, 2, and 3). As an example, according to the required procedures, the immediate predecessors of *Closed* are either *Authorized* or *Declined*. The mined model indicates instances where neither was included in the trace preceding *Closed* (e.g., *Draft* ->*Closed*, or *Draft* ->*Auth Process* ->*Closed*). These are classified as workarounds of Type A.

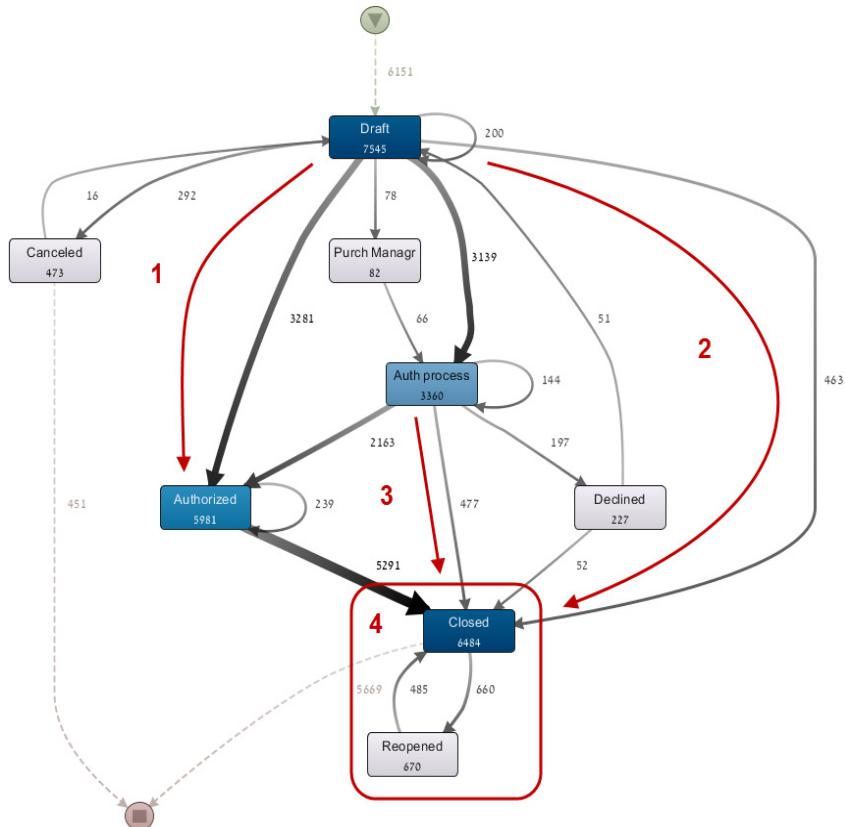


Fig. 1. Bypassing process steps in a purchase requisition approval process (Case Study 3)

Type B – Selecting an Entity Instance That Fits a Preferable Path:

Process instances where this type of workaround is committed are legitimate instances in terms of control flow. In fact, they might seem legitimate in every process aspect. Yet, they are not accurate reflections of the real life process. Hence, mining event logs cannot detect workarounds of this type. When specific selection types are known to exist through domain knowledge (e.g., splitting purchase requests), it might be possible to formulate identifiable patterns that would help quantifying these specific behaviors, but these would not be applicable for discovering other cases of this type. It might be possible that data mining techniques aimed at fraud detection (e.g., [17]) can be used for this purpose. However, this is beyond the scope of this paper.

Type C – Post Factum Information Changes:

Workarounds of this type take place at certain stages of the process. Specifically, update data operations are performed after the data has been used by decision making steps (e.g., approval). However, not every modification in the value of a data item that takes place after the data has been used is illegitimate (e.g., errors can be identified and corrected). For a data update to be considered workaround of this type, three conditions should hold:

- (1) An update is performed to a data item that has been used previously in the process.
- (2) The previous use was for decision making.
- (3) After the data update, the process instance does not iterate back to the decision making step (for revisiting the decision based on the updated value).

Clearly, these conditions cannot be directly checked in an event log without additional domain knowledge that would indicate which data is used for decision making at which process steps. Without such indication, skipping reiteration after the update of the data would appear like bypassing process steps (workaround type A).

Using domain knowledge, we can identify data update activities and decision activities relying on the relevant data item.

Consider a data update activity u , and let d be an activity where this data is used as a basis for decision making. Assume u appears in a given trace in the i th position, while d can appear in position j , $j < i$. If d is not included in the trace in position k , $k > i$, then this trace includes a workaround of type C.

Note that more than one decision activity might be needed according to the process definition. It should be possible to similarly check the existence of several activities in the remaining part of the trace (at least one or all together).

Area 4 in Fig. 1 provides an example of post factum updates, where purchase requisitions that are already closed are reopened for updating their data (update activity) and then closed again. One related decision activity that should follow reopening is *Authorized*. In the mined process, 485 of the 660 instances that were reopened were then closed (while the remaining ones, which reiterated to approval steps, have been filtered out in the analysis).

Type D – Incompliance to Role Definition: these workarounds are characterized by situations where participants perform activities outside the realm of their responsibility. Apparently, it is easy to detect such workarounds by comparing the user of every activity with the list of users who are permitted to perform it. However, these workaround can only take place if the permissions defined in the system are not tight enough, so unauthorized users can perform the activities. Hence, for accurately detecting these workarounds, the permission assignment should be prepared by the process owner independent of the existing system permissions. Based on such list, identifying activities that are performed by unauthorized users is straightforward.

Denoting the set of users who are authorized to perform activity a by $AT(a)$, and consider a trace where a is performed by user u . If $u \notin AT(a)$ then the trace includes a workaround of type D.

As an illustration, Table 1 presents the authorized and actual users of activities in a process taken from one of the organizations that were studied. As can be seen, some activities are performed by unauthorized users. In particular, financial approval (3022 out of 3326 times performed by the user P9) and final approval (3065 out of 3303 times performed by P11) are performed by several other users who are not authorized to perform them.

However, it might be that a temporary permission has been granted to, e.g., P8, to perform these activities when the employee responsible for them was away. If that is the case, then along the time, the instances where P8 performed these activities should appear in one or several relatively short periods. This was not found in our case, where the instances involving P8 in these activities were scattered along the two years whose logs were analyzed.

Table 1. Authorized vs. actual users of activities

Activity	Participant												Total
	Auth	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	
Create PR	all	454	1185	0	223	1	0	175	343	44	1263	0	3688
Manager approval	P1 P10	376	0	0	0	1	0	0	0	0	1121	0	1498
Financial approval	P9	0	38	170	35	16	0	0	44	3022	1	0	3326
Director approval	P11	0	0	0	0	0	0	0	0	0	0	190	190
Buyer approval	all	0	1119	1308	160	26	0	169	311	3	0	0	3096
CEO approval	P5	0	0	0	0	3307	0	0	0	0	0	2	3309
Final approval	P11	1	13	0	2	0	96	9	102	0	15	3065	3303
Cancel PR	all	11	30	9	1	3	0	20	5	3	8	0	90
Close PR	all	356	1109	1	184	0	0	163	341	3	1125	0	3282
Reopen PR	all	0	0	0	0	0	0	0	10	0	1	0	11
Total		1198	3494	1488	605	3354	96	536	1156	3075	3534	3257	21793

Type E – Fictitious Entities: in workarounds of this type, a fabricated entity instance is created, to allow the users manage and document process parts that are not included in the formal process (and hence cannot be properly monitored and documented). The resulting process instances appear like legitimate process instances (although they would typically not cover the entire process, but only specific parts).

Following this, mining the control flow of the process would not provide any indication of these workarounds. However, employees who perform workarounds of this type typically mark the fictitious entities by specific codes, so they can distinguish them from real ones. For example, in the student intake process described above, fabricated students were always assigned to Room 1000 (which was fabricated too). If this "marking" information is provided by a domain expert, the relevant process instances can be identified, but this would only serve for quantification of a known phenomenon, not for discovery of unknown ones.

Type F – Separation of the Actual Process from the Reported One: these workarounds entail manual performance of process parts (which cannot be reflected in the log), and reporting the actions to the system just for the record, at some unrelated time. While we cannot tell what actually took place in the (manual) process, the post-hoc recording would usually reflect a "normal" and legitimate process execution, compliant with the required procedures.

Still, we suggest that at least some of these workarounds can be tracked by situations of substantial delays in the process, immediately followed by a bundle of transitions appearing one after the other in an unreasonably short time (as compared to the "normal" process transition times, e.g., three activities performed within a few minutes). For example, consider the instance of a purchase requisition approval process depicted by the log in Table 2. The activity of *Director approval* takes an extremely long time (compared to the activities that precede it), and is followed by two activities whose duration is less than one minute. It is reasonable to believe that the process has in fact progressed before the *Director approval* has been formally given, and that *Approve PR* and *Close PR* are just reported in a post hoc manner.

Table 2. An example log part demonstrating workaround type F

Activity	Date	Start Time	Duration
Create PR	11.10.2011	12:27:00	9 mins
Buyer approval	11.10.2011	12:36:00	2 hours, 52 mins
Financial approval	11.10.2011	15:28:00	6 hours, 11 mins
CEO approval	11.10.2011	21:39:00	10 hours, 10 mins
Director approval	12.10.2011	07:49:00	15 days, 46 mins
Approve PR	27.10.2011	08:35:00	< 1 min
Close PR	27.10.2011	08:35:00	< 1 min

It can hence be concluded that instances including workarounds of this type might seem legitimate in terms of their control flow, but can be detected based on activity durations. For each activity a , we need to define an upper duration threshold $UDT(a)$ and a lower duration threshold $LDT(a)$.

For a given trace, if two consecutive activities a and b are found, such that their durations satisfy $d(a) \geq \text{UDT}(a)$, and $d(b) \leq \text{LDT}(b)$, then the trace includes a workaround of type F.

The duration thresholds can be defined based on the log, e.g., by setting a range such that the durations of a defined ratio of the activity instances in the log are above (or below) that range. Note that the upper threshold might even be slightly above the average duration, but the lower threshold needs to be such that the activity cannot possibly be executed within this time. Often, there would be several activities, whose durations are below the lower threshold, performed one after the other. These would be all the activities that have been performed off-line and reported in retrospect.

4 Application to Real Logs

The previous section provided means for identifying four of the six workaround types in event logs. This section reports the results obtained for logs of five processes taken from three organizations over two years. We aimed at addressing processes whose roles are similar in different organizations, as detailed in Table 3.

Table 3. Processes whose logs were analyzed

Process	Title	Organization description
1	Purchase requisition approval	Academic organization, 500 employees
2	Purchase requisition approval	Manufacturer of control and monitoring systems, 300 employees
3	Purchase ordering	Marketing organization, importing and selling medical equipment, 300 employees
4	Purchase requisition approval	
5	Purchase ordering	

As discussed in the introduction, we have decided to use Fluxicon Discovery as a commercial process mining platform, currently available to organizations. The conditions discussed in Section 3 were operationalized using the necessary domain knowledge which was obtained from the process owners. The detailed conditions were then implemented as separate filters over the event logs. Table 4 provides the findings that were obtained. Note that each workaround type was addressed separately, so summarizing all types together would not make sense, since there are instances where more than one workaround type was detected. Moreover, some workarounds can be classified to more than one type. For example, when workarounds of type F (actual process vs. reported one) are performed, often the same person reports several operations, including ones outside his/her role (thus they can also be classified as workarounds of type D).

As can be seen in Table 4, organizations as well as processes within the same organization differ from one another in the frequency of workarounds and in their types. In general, workarounds of type A (bypassing) are the most frequent ones. Difference among organizations is especially evident with respect to organization 1, whose number of workarounds is extremely low in the purchase requisition approval process. In contrast, in the other two organizations the purchase requisition approval

process has a much higher workaround rate than the purchase ordering process. In organizations 2 and 3 the requisition approval process entails a high number of workarounds, especially of type A (bypassing). In organization 2, type D (incompliance to role definition) is also frequent, and in organization 3, types F (actual vs. reported process) and C (post factum information change) are often taken.

Table 4. Workaround percentage by type

Organization	Process	Number of instances	% instances with workarounds by type			
			A	C	D	F
1	PR approval	3688	5.1%	1.3%	2.7%	5.9%
2	PR approval	6920	53.2%	8.8%	22.3%	12.0%
	Purchase ordering	4211	6.8%	7.2%	24.4%	12.6%
3	PR approval	21289	75.3%	25.0%	3.5%	68.1%
	Purchase ordering	5217	11.9%	4.8%	9.0%	4.1%
Average in all processes			30.5%	9.4%	12.4%	20.5%

We note that considering our notion of workarounds, these findings might include both false positives, cases that are falsely indicated as workarounds, and false negatives, actual workarounds that are not detected. Specifically, we define workarounds not just as incompliant behavior, but as one that involves intentional defiance of known procedures. Clearly, we have no means for assessing user intention from event logs. To this end, we rely on the list of workaround types, which was obtained through interviews where users indicated what they perceive as workarounds. It might be that the resulting patterns also include incompliant behavior performed for different reasons.

For example, the cases identified as workarounds of type C (post-factum information change), might include error corrections (where data should be modified to correct the error). According to the regulations, re-iterations to the decision steps (e.g., approval) were required. It might be that this was done informally by emails or phone calls, but the system has no track of these. Hence, officially these cases are considered as workarounds. Similarly, identified cases of type D (incompliance to role definition) might include cases where a temporary permission was granted by the authorized user. We tried to detect these cases by examining the distribution of these occurrences over time. However, one-time permissions cannot be detected this way.

False negatives would relate mainly to types A (bypassing) and F (actual process vs. reported one). Bypasses (type A) can be performed manually (e.g., ordering goods by phone) and not reported, while the process as reflected in the log appears to progress according to the required procedures. Considering separation of the actual process from the reported one (type F), our detection method is based on the assumption that this can be reflected in the log as exceptional durations of activities (exceptionally long duration of one activity followed by one or more exceptionally short durations). This assumption does not necessarily apply in all cases. Specifically,

the post-hoc reporting might be performed at different points in time for different activities, which would not appear as exceptional activity durations.

Still, even with these limitations, we believe that quantification like the one in Table 4 is valuable for organizations. In particular, it can serve as a starting point for investigating the workarounds that are performed and lead to corrective actions that should address the reasons that drive these workarounds. The result of such actions should be improved processes with improved compliance.

Finally, we note again that two types of workarounds were not possible to detect from the logs, yet they are likely to exist. Being aware of this possibility, organizations can apply targeted means for identifying and addressing them. Fictitious entities (type E), for example, usually involve practices which are well known among the relevant users, sometimes even anchored in departmental documents and procedures. Typically, they are marked by specific IDs that would enable the users to track them. It should hence be rather easy to specifically elicit them from the employees and make appropriate modifications to the process. Intentionally selected entity instances (type B) would be more difficult to expose, especially since these are performed by sophisticated employees with the intention to avoid the required process paths. As discussed, data mining techniques might be of assistance.

5 Related Work

While much attention has been given to compliance management in general and compliance checking in particular, the specific phenomenon of intentional workarounds has not been extensively investigated. Nevertheless, the conditions defined here in correspondence with workaround types can be verified by some of the existing compliance checking approaches. This section reviews the relevant literature, indicating the workaround types that can be detected by each approach.

Several approaches have been suggested for backward compliance checking. These include replaying-based techniques (e.g., [21][4][6][7][8]), where a process is replayed on the log against the required process model, and rule checking techniques, where rules can be defined using Linear Temporal Logic (LTL) [3][13] or Petri net representation [9][10][14][15][18]. Replay-based techniques address incompliant behavior in general, as opposed to the specific set of behaviors we address in this paper. Behavior types that would be detected by these techniques include some of the workaround-related patterns, as well as additional ones, such as activity repetition, or performance of additional or different activities as compared to the process definition. In contrast, rule-based conformance checking can relate to specifically defined rules (including those related to workarounds). Hence, we focus on this group of approaches.

[18] define 15 categories of control-flow compliance rules. Four of these categories are relevant in our context of workaround detection. *Existence rules* limit the occurrence or absence of a given event within a scope – these can be used for identifying workarounds of type A (bypassing) and of type C (post-factum information change). *Precedence rules* require or limit the occurrence of a given event in precedence to another event – these can be useful for detecting workarounds

of type A, since a violation of such rule implies that activities have been bypassed. *Response rules*, which require or limit the occurrence of a given event in response to another event – can be used for detecting workarounds of type C, where a post-factum information change is considered as workaround only if it is not followed by reiteration of decision steps. *Between rules* require or limit the occurrence of a given event between two other given events – can be used for detecting bypassing (type A) in a process which includes loops.

These compliance rules can be checked by LTL-based approaches [3][13], which are easily capable of specifying these kinds of constraints. Petri net-based methods specify a rule as a Petri net segment, and then find a best alignment with the log [2][18][19]. While LTL-based rules address only the control flow of the process, and are thus relevant for detection of the two aforementioned workaround types (A and C), the alignment seeking Petri net based approaches can handle other aspects as well.

[18] address compliance to data and organizational aspects, which enables detecting workarounds of type D (incompliance to role definition). The data-related techniques are extended in [19] to address temporal constraints, which are capable of capturing the exceptional activity durations that characterize workarounds of type F (actual vs. reported process). It can hence be concluded that the alignment-based methods provide powerful means that enable specifying appropriate rules and detecting the four workaround types that are reflected in event logs.

6 Conclusion

Workarounds are often performed in business processes. Compliance management literature has not addressed them as a distinct phenomenon so far, but rather as part of incompliant behavior in general. We believe that intentional defiance of known procedures should receive special attention, since revealing this behavior and the reasons that motivate it can expose many underlying problems that need to be solved.

A main contribution of this paper is in approaching this issue from a practice perspective. As opposed to existing works in the area of compliance checking, which focus on the capabilities of technology to be utilized, this paper departs from behavior types that exist in practice and are perceived by employees as intentional workarounds. It uses six generic behavior types identified in organizations, and seeks for technological solutions that can serve for detecting these behaviors. It does so by analyzing and characterizing the log patterns that can be associated with the considered workaround types. We have specified conditions that enable detecting four workaround types in event logs and demonstrated their ability to quantify the occurrence of each type in logs of five real-life processes.

An important finding is the indication of two workaround types that leave no recognizable trace in the log and hence cannot be generically identified by process mining techniques. Still, additional domain knowledge can be used for defining specific patterns that might be identified in logs. This highlights the limitations of generic process mining techniques and can guide organizations in further directions that need to be taken to completely address the workaround phenomenon.

Developing an understanding of the workarounds that take place and particularly of the reasons that drive them would be valuable in improvement efforts. Corrective actions can include redesigning the processes, improving the data flow, the permission and control mechanisms, role definitions, and also training and disciplinary actions. This is expected to lead to improved performance as well as compliance.

Future research will aim at investigating the reasons for workarounds, and establish relationships between process properties, such as bottlenecks and number of participants, and the frequency of workarounds.

References

- [1] van der Aalst, W.M.P.: *Process Mining – Discovery, Conformance and Enhancement of Business Processes*. Springer (2011)
- [2] van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. *WIREs Data Mining Knowledge Discovery* 2, 182–192 (2012)
- [3] van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.F.: Process mining and verification of properties: An approach based on temporal logic. In: Meersman, R., Tari, Z. (eds.) *CoopIS/DOA/ODBASE 2005*. LNCS, vol. 3760, pp. 130–147. Springer, Heidelberg (2005)
- [4] van der Aalst, W.M.P.: Business Alignment: Using Process Mining as a Tool for Delta Analysis. In: Grundspenkis, J., Kirikova, M. (eds.) *Proceedings of BPMDS 2004*. Caise 2004 Workshops, vol. 2, pp. 138–145 (2004)
- [5] Abdullah, N.S., Sadiq, S.W., Indulska, M.: Information systems research: Aligning to industry challenges in management of regulatory compliance. In: PACIS 2010, p. 36. AISeL (2010)
- [6] Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Towards Robust Conformance Checking. In: Muehlen, M.z., Su, J. (eds.) *BPM 2010 Workshops*. LNBP, vol. 66, pp. 122–133. Springer, Heidelberg (2011)
- [7] Adriansyah, A., Sidorova, N., van Dongen, B.F.: Cost-based Fitness in Conformance Checking. In: ACSD 2011, pp. 57–66. IEEE (2011)
- [8] Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: EDOC 2011, pp. 55–64. IEEE (2011)
- [9] Calders, T., Guenther, C., Pechenizkiy, M., Rozinat, A.: Using Minimum Description Length for Process Mining. In: SAC 2009, pp. 1451–1455. ACM Press (2009)
- [10] Fahland, D., de Leoni, M., van Dongen, B.F., van der Aalst, W.M.P.: Conformance Checking of Interacting Processes with Overlapping Instances. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) *BPM 2011*. LNCS, vol. 6896, pp. 345–361. Springer, Heidelberg (2011)
- [11] Kharbili, M.E., de Medeiros, A.K.A., Stein, S., van der Aalst, W.M.P.: Business process compliance checking: Current state and future challenges. In: MobIS 2008. LNI, vol. 141, pp. 107–113. GI (2008)
- [12] Kharbili, M.: Business process regulatory compliance management solution frameworks: A comparative evaluation. In: APCCM 2012. CRPIT, vol. 130, pp. 23–32. ACS (2012)
- [13] Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative Specification and Verification of Service Choreographies. *ACM Transactions on the Web* 4(1), 1–62 (2010)

- [14] Muñoz-Gama, J., Carmona, J.: A Fresh Look at Precision in Process Conformance. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 211–226. Springer, Heidelberg (2010)
- [15] Munoz-Gama, J., Carmona, J.: Enhancing Precision in Process Conformance: Stability, Confidence and Severity. In: CIDM 2011. IEEE (2011)
- [16] Outmazgin, N.: Exploring Workaround Situations in Business Processes. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 426–437. Springer, Heidelberg (2013)
- [17] Phua, C., Lee, V., Smith-Miles, K., Gayler, R.: A comprehensive survey of data mining-based fraud detection research. Artificial Intelligence Review (2005)
- [18] Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where Did I Misbehave? Diagnostic information in compliance checking. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 262–278. Springer, Heidelberg (2012)
- [19] Ramezani, E., Fahland, D., van Dongen, B., van der Aalst, W.M.P.: Diagnostic Information for Compliance Checking of Temporal Compliance Requirements. In: Proceedings of CAiSE 2013. Springer (2013)
- [20] Ramezani, E., Fahland, D., van der Werf, J.M., Mattheis, P.: Separating compliance management and business process management. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011 Workshops, Part II. LNBIP, vol. 100, pp. 459–464. Springer, Heidelberg (2012)
- [21] Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)

Using Data-Centric Business Process Modeling for Discovering Requirements for Business Process Support Systems: Experience Report

Ilia Bider^{1,2}, Erik Perjons¹, and Zakria Riaz Dar¹

¹DSV, Stockholm University, Stockholm, Forum 100, SE-16440 Kista, Sweden

²IbisSoft AB, Stockholm, Box 19567, SE-10432 Stockholm, Sweden
ilia@{dsv.su, ibissoft}.se, perjons@dsv.se, zrdar@kth.se

Abstract. Building a process model is a natural part of the requirements engineering (RE) when creating requirements for a computerized system/service to support a business process. When the process in question is workflowable (i.e. a process in which the order and the flow of tasks/operations/activities can be predefined), there are plenty of modeling techniques, notations and tools that can help in this undertaking. These techniques, however, are of little use for discovering requirements for support of non-workflowable processes in which the information artifacts created in the process (e.g. reports, lecture slides, budget documents) are of more importance than the flow of tasks/operations/activities. Other types of techniques, notations and tools are required in this case. This paper reports on a project of using a data-centric modeling approach supported by a computerized tool in RE. The goal of the project was to test whether the approach could be useful for the task of discovering requirements on a computerized system/service supporting the process, and which and how much of requirements could be captured using it. The process used in the test is a process of course preparation in the authors' own department. The paper reports on the environment in which the project has been conducted, results achieved, and lessons learned.

Keywords: Requirements Engineering, RE, Requirements discovery, business process, data-centric.

1 Introduction

Following Ian Alexander [1], we consider that all important requirements cannot be gathered from stakeholders directly, but need to be discovered, which warrants using special techniques and tools different from the ones used for managing already discovered requirements. As our concern is requirements for computerized systems/services that support business processes, discovering details of the process to support is an essential part of the requirements discovery.

A systems/services can be aimed at supporting an already existing process, or a process that needs to be designed or improved alongside with developing a support system. Independently of which of the above is the case, it is people who are (will be)

engaged in the process who have relevant tacit knowledge that needs to be unearthed during the discovery of requirements. Therefore, the role of techniques and tools used in the requirements discovery is to facilitate the existing or future process participants to reveal the tacit knowledge they possess. According to [1], techniques and tools for requirements discovery should be quite simple so that the focus will not be moved from discovering requirements to designing the system.

When there is a good chance that the process to be discovered has a strict order of tasks/operations/activities, usual process modeling techniques based on the workflow-thinking could be tried in the discovery process. These range from simple charts to complex workflow diagrammatic languages such as BPMN 2.0, and they are supported by a number of modeling tools. However, when the chances that the process will be workflowable¹ are small, these techniques and tools might not be appropriate, and other means should be engaged in the requirements discovery phase.

In this paper, we consider the problem of discovering requirements for processes in which information/data processing by collaborative teams constitutes the core of the process. In addition, we do not require such process to be workflowable. We believe that for this kind of processes, a data-centric process modeling technique is more appropriate as far as process discovery is concerned.

In this paper, the term data-centric modeling is understood in a broad meaning. Namely, as data-centric we consider any process modeling technique that permits to start structuring data/information processed in the frame of the process before the details of the flow of tasks/operations/activities are known. To this category, for example, belong artifact-based modeling [3], data-driven modeling [4] and state-oriented modeling [5]. Defining folder structures for case-based systems [6] could also be considered as belonging to the data-centric process modeling².

The goal of the project reported in this paper was to investigate whether a data-centric modeling technique supported by a computerized tool is suitable as a means for discovering requirements for business process support (BPS) systems/services. More specifically, we aimed at getting answers to the following **three questions**:

1. Whether such an approach is suitable for use in requirements discovery facilitating workshops.
2. Whether the requirements discovered in the workshops could be represented in a form suitable for discussing them with a broader audience that includes stakeholders who have not participated in the facilitating workshops.
3. Which and how much of requirements could be discovered with this approach .

Our search of the works related to the above questions produced no results, thus, to the best of our knowledge, the current work is the first attempt to get answers to these questions³.

¹ As workflowable, we consider a process where the order and the flow of tasks/operations/activities can be predefined. For more exact definition of workflowability see [2].

² The main difference between a data-centric and traditional workflow process modeling is that in the former the focus is on information artifacts, e.g. reports, lecture slides, budget documents, while in the latter the focus is on operations/activities that produces the artifacts.

³ Our past experience of state-oriented process modeling [4] lacked proper tool support.

Investigation was conducted in the frame of a real organization – department of Computer and Systems Sciences at Stockholm University. Though the study has been conducted only in one organization, there is a likelihood that the results achieved are of general nature; based on the authors' previous experience (see some examples in [5]) this particular environment is quite typical for non-workflowable processes.

The rest of the paper is structured according to the following plan. In Section 2, we describe the project settings that include short description of the organization, business process under investigation (course preparation process), and project team. Section 3 describes our efforts to find a tool to use in the project. Section 4 overviews the tool used in the project. Section 5, describes completion of the project. Section 6 discusses the model built during the project. Section 7 discusses the results achieved and lessons learned. Section 8 contains concluding remarks and plans for the future.

2 The Project Environment

2.1 The Organization

The project has been completed in the department of Computer and System Sciences, abbreviated to DSV, at Stockholm University. The department is engaged in research and undergraduate and graduate teaching of about 5 700 students simultaneously. It runs bachelor, master, and doctoral programs in the fields of Computer Science and Information Systems. It has about 280 staff members including teachers and administrative personal. It also has its own IT department that operates department specific software, while the general software is operated by the central IT unit of Stockholm University.

The IT department, besides operating the software acquired from various vendors, has its own development unit engaged in developing department specific software. The latter includes development of systems that supports teaching and learning. The unit does not have a strong tradition of requirements engineering, which results in long cycles of getting the system and it users synchronized.

Modern process modeling tools are not used during requirements engineering phase in the department. The systems that support teaching and learning, from outside vendors and from own development, are not of the type of process aware systems. They have quite reach functionality, but the information on when and how to use the functions included in these systems resides mostly in the heads of their users.

2.2 The Process

The business process chosen for the study is the process of preparing a course occasion to be given by the department. The occasion can be the first occasion of a completely new course, or just an ordinary occasion of a regularly given course. This business process has been chosen based on the following two reasons:

- It is a typical process in the department.
- The process does not have real computerized support. The results of it are to be placed in different systems, e.g. lection slides needs to be made available to the students for download. However, these systems do not support course preparation.

Below, we present an informal overview of the chosen process. In this overview, we also include activities related to giving and evaluating the course. We identify five major phases in the course process⁴:

1. *Planning course* includes a number of meetings with involved teachers to decide which teaching and learning activities to carry out during the course as well as their sequence. The phase also includes deciding and producing the course material for the course. Finally, an evaluation form needs to be designed, which will be filled out by the students when the course has ended.
2. *Schedule course* consists of composing a schedule with dates, times and locations for the lectures, lessons and seminar as well as a date, time and location for the written exam and other teaching and learning activities. The phase includes a number of interactions between the teacher responsible for the course and the person responsible for scheduling courses in the department.
3. *Publishing course material* consists in printed course material. The printing is done by the person responsible for printing.
4. *Learning and teaching* includes a number of teaching and learning activities, such as lectures, lessons and seminars, managing assignments and carrying out exams. It also includes giving feedback on and/or grading reports and exams.
5. *Evaluation* includes the students evaluating the course after the end of the course. The phase also includes an analysis of the evaluation carried out by the teacher responsible for the course.

2.3 The Team

The project involved four teachers and one MS student; this group will be referred to as the *extended group*. The major team consisted of two teachers and one MS student (all authors of this paper); this group will be referred to as the *modeling group*. One of the teachers had long experience of giving courses in the department, the other one was a novice. The student represented the "learning" stakeholders. The additional two teachers, referred to as the *external domain experts*, had long experience of teaching in the department. They participated only in the evaluation of the approach's results. They were not involved in requirements discovery, and knew nothing about the project beforehand.

3 Selecting a Modeling Tool

3.1 Requirements on a Tool to Be Used

We were looking for a data-centric process modeling tool or a BPM suite of this kind that could be suitable for performing multiple Requirements Engineering (RE) tasks

⁴ Though the process is split in a number of phases, the latter are not being executed in a sequence but can run in parallel (see Section 6), which makes the process non-workflowable according to [2]. Full analysis of workflowability of this process is not presented due to the size limitations, but will be published elsewhere in connection to another topic.

[1]. In particular, we looked for a tool that could be used directly in facilitating workshops for discovering the following aspects of the process to be supported:

1. *Structure of data/information* created and utilized in the process
2. *Data/information flow* in the process
3. *Participant collaboration* in the frame of process instances
4. *Categories of users* engaged in the process and limitation on the data/information they can access (read, write, or modify)
5. *Operations/activities* included in the process and *restrictions* on the order in which they can be completed

Additionally, we preferred a tool/suite to be suitable for:

- Designing a *prototype* of the system to give future users some understanding of what would it mean for them to run a process supported by the system to-be
- Discussing and recoding process *scenarios* based on the past experience (process cases)

In addition, we preferred a domain independent tool that could be used for various kinds of processes, general administration, teaching and learning, research, etc.

3.2 Searching for a Tool

Right from the project start we had one candidate for a tool to be used in the project, namely, a cloud-based service called *iPB* [7,8], developed by *ibisSoft* with which the first author was associated. *IbisSoft* had the policy of providing a limited demo-license for research purposes free of charge, so it was easy to obtain access to the tool. Though *iPB* had not been explicitly developed as a data-centric modeling tool/suite, our preliminary analysis showed that *iPB* satisfied the requirements set in the previous section.

Despite having a candidate, we decided to spend some time looking for other candidates to be used, in case we can find a better alternative. In preparation for the tool selection, we created a list of criteria for tool evaluation. This list is based on the requirements from the previous section and general properties of modeling tools from the literature, see, for example, [9]. The list includes the following criteria:

- *Availability*. Firstly, the tool should be available for usage, e.g. commercially available, or as an open source. Secondly, it should be easily accessible from any place one can possibly need to have access to it [1]. For our purpose, it would be desirable to have a web-based/cloud-based tool.
- *Domain-independence* (expressiveness or universality in terms of [9]). The tool should be possible to use in different application domains.
- *Completeness* [9]. The tool should have means to express all concepts considered to be important for the given objective of modeling.
- *Comprehensibility* [9]. The models, even the intermediate ones, should be easy to comprehend for domain specialists without prior knowledge of business process modeling.

- *Tasks flexibility*. The tool should be possible to use for different types of activities like modeling, prototyping, scenario testing (see the list in the previous section).
- *Tasks suitability* - extension of task flexibility. The tool should be suitable for the tasks for which it has been chosen [9], i.e. allows completing them in a convenient way.
- *Usage flexibility* (or arbitrariness in terms of [9]) – extension of task suitability. The tool should not impose hard restriction on its usage, but gives the user freedom to choose how to use it.
- *Coherence* [9]. Different components produced with the help of the tool should be integrated to constitute a whole.

When searching for a tool, we did not have in mind finding the best possible candidate that satisfies the criteria, the first good enough choice would be sufficient for us. The goal of the whole project was to test a data-centric approach of business process modeling supported by a tool/suite. Which tool to use was considered of lesser importance.

Through the quick search on "data-centric" and "artifact-based" process modeling we found a number of research articles, but only two references to potential tool candidates, both from IBM - FastPast and Siena described in [3]. Both were experimental tools that were supposed to be available for research and education purposes, but we found no URL with instructions on how to get access to them.

As our initial efforts to find an available tool through a general search were unsuccessful, we decided to stop the search and use the tool already at our disposal.

4 The Modeling Tool Described

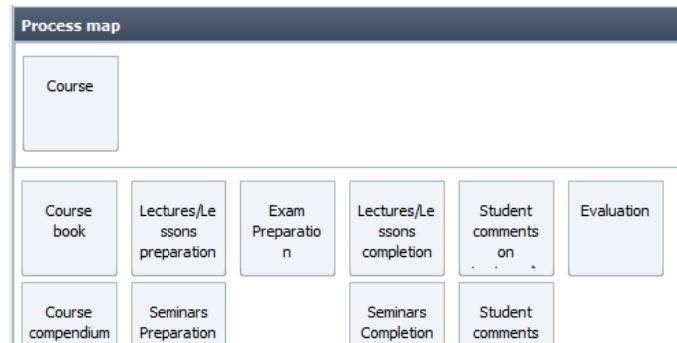
4.1 *iPB* as a Data-Centric Business Process Modeling Tool

iPB [7,8] was designed as a tool for developing BPS systems/services for loosely structured business processes. One part of such development consists of designing a process model that the tool interprets at runtime while providing support for process participants.

iPB consists of two components - *Design studio*, and *Runtime environment*. Design studio is used for building a process model, while Runtime interprets this model helping process participants to run their process instances/cases.

Process modeling in *iPB* is based on four *main* abstract concepts: *Process map*, *Process step*, *Step form*, *Form field* (additional concepts are described later) The basic relationships between these *main* concepts are as follows.

- A *process map* consists of a collection of named *process steps* arranged on a two-dimensional surface called *process layout*. The layout consists of two areas – (a) the upper row called *flow-independent* steps, and (b) a low area, a two dimensional matrix called *flow-dependent* steps, see Fig. 1.
- Each *process step* in a *process map* has a *step form* attached to it.

**Fig. 1.** Process map

- A *step form* consists of a collection of named *form fields* arranged in a two-dimensional matrix called *form layout*, see Fig. 2. Each field belongs to a certain type. There are simple types, like, text, multi-text, date, date-time, option list, checkbox (boolean), etc., and complex types, like uploaded file, journal, person, organization. In addition, field collections that belong to different step forms can intersect. This is done by defining fields in one form, and then referring to them in other forms.

Step Form Grid					
Title	Type	Mandatory?	Teacher 1	Teacher 2	Location
Start	Finish		Teacher 1	Teacher 2	Location
ALTERNATIVE SCHEDULE					
Start	Finish		Teacher 1	Teacher 2	Location
Description	Lecture Presentation				Other Info
Forum for discussing under ...					

Fig. 2. Step form for step Lectures/Lesson preparation from Fig. 1

A process map with underlying step forms defines a kind of a database scheme for the given process type/template. This scheme is then used by the runtime for "creating" a database for each instance/case of the process, started based on this process map. The runtime system also interpret step forms as web forms for inputting, viewing and changing information in the instance database, see Fig. 3. The process map is used for

creating an *instance map*, see Fig 4, that serves as a mechanism for user navigation through the web forms. To open the web-form, the user just clicks on the step in the *instance map*. As we see from Fig. 3, some steps are allowed to have multiple forms at runtime (see the tabs for each lecture in Fig. 3).

The screenshot shows a web-based form for managing a lecture step. At the top, there is a navigation bar with tabs for 'Lecture 1' (selected), 'Lecture 2', 'Lecture 3', 'Lecture 4', 'Lecture 5', 'Lecture 6', 'Lecture 7', and 'Lecture 8'. Below the tabs is a toolbar with icons for 'New form', 'Change form name', 'Write protect', 'Delete form', and 'Print'. The main content area is divided into several sections:

- Title:** Kursintroduktion
- Type:** Lecture, Lesson
- Mandatory?** Mandatory, Optional, Strongly recommend
- Start:** 2012-10-04 08:00
- Finish:** 2012-10-04 09:45
- Teacher 1:** ErikP - Perjons, Erik
- Teacher 2:** KarlP - Petersson, Karl
- Location:** Sal A
- ALTERNATIVE SCHEDULE** (In case of many students):
 - Start:** 2012-10-04 14:00
 - Finish:** 2012-10-04 15:45
 - Teacher 1:** -- Choose --
 - Choose --
 - Teacher 2:** -- Choose --
 - Choose --
- Description:** Tahoma

Course Introduction of ITO and its objectives
- Lecture Presentation:** IntroITO.pdf
 - [Open](#)
- Forum for discussing under preparation:**
 - [Add](#)
 - [Edit](#)
 - [Copy](#)
 - [Delete](#)
 - [Print](#)

Date	By	Comment
2012-12-14 23:18	Ilia - Bider (Admin), ilia	Please have a look on the slides and give your opinion
		By Ilia - Bider (Admin), ilia 2013-02-23 23:19

Fig. 3. Step form from Fig. 2 as a web-form

Besides the *main* concepts described above, *iPB* provides additional capabilities, e.g.

- *Business rules* that establish relationships between the steps. One type of such rules controls whether the user can open a particular step form based on the state of completion of other forms. Steps that cannot yet be clicked on are colored grey, see Fig. 4. Such rules are specified with the help of a square matrix where both rows and columns correspond to the steps defined in the process. In this matrix, the content of a cell can determine that the row step can be started only after the column step has been completed (blue color in Fig. 4), or started (green color in Fig. 4). Other types of rules prescribe synchronization of steps with multiple forms, e.g., steps "Lectures/Lessons preparation" and "Lectures/Lessons completion" in Fig. 1 and 4 (for more on synchronization see Section 6).
- *Business rules* that establish when data in a step form can be saved or the step can be closed. Such rules are expressed via defining some fields on the form as mandatory for save, or close.
- *User profiles* that specify categories of users and their access rights to read, and change information in each step form.
- *Visual properties* assigned to the fields that change their look and feel at runtime.
- *Step properties*, such as permission to have more than one form (see Fig 3).

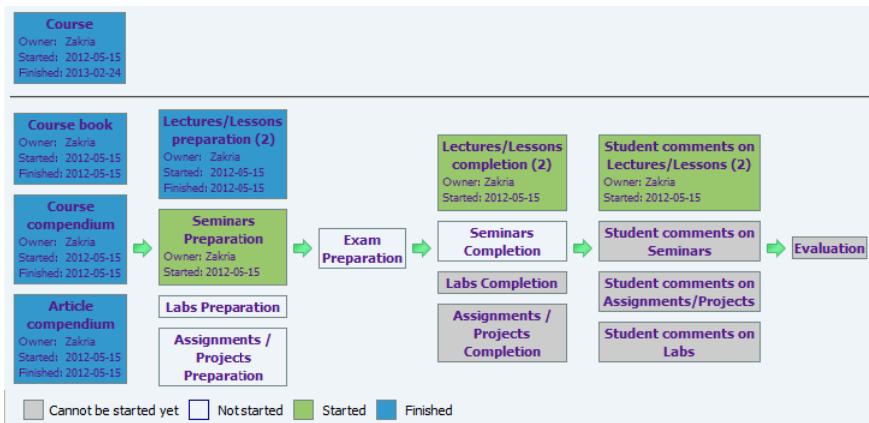


Fig. 4. Instance process map from Fig. 1 as a user navigation mechanism

4.2 Business Process Modeling with *iPB*

In section 3.1, we listed aspects of a business process that we wanted to be depicted in the model, and additional tasks that the modeling tool chosen for the project could help to perform. Below, we show how these aspects and tasks are covered by *iPB*.

- *Structure of data/information* created and utilized in the process is defined through creating step forms.
- *Data/Information flow* in the process is defined by including references to the form fields from the previous step forms into the form where this information is used.
- *Participant collaboration* in the frame of process instances is defined by inserting complex fields of the type *Journal* in the step form; see the field at the bottom in Fig. 2 and 3. Users working with the same form add records to the journal registering questions, answers, comments, etc. relevant to the particular step.
- *Categories of users* engaged in the process and limitation on the data/information they can access (read, write, or modify) are defined with the help of user profiles.
- *Tasks/operations/activities* included in the process and *restrictions* on the order in which they can be completed are specified in the following manner. The concept of step is used for representing larger chunks of work - work-packages; if smaller operations are needed they are represented in the step forms as checklists in the way as accepted for case-management/adaptive case management systems [6]. The order of steps is represented by (a) process layout that gives a recommended sequence of steps (from left to right and from top to bottom), and (b) by business rules that restrict the possibility to “jump over” some steps. On the lower level, making operation checklists mandatory gives an opportunity to prevent finishing the step to which they belong until all operations have been completed. The latter can prevent starting the next step if it is forbidden by business rules that require finishing the previous step first.

- As far as *prototyping* is concerned, the *iPB*'s runtime system automatically creates a system prototype that can be tested by the future end-users. By using visual properties of fields, there is a possibility even to design the exact layout of the forms to be used in the future system.
- The run-time system allows also to record and discuss *scenarios* of the past process instances/cases.

Note that the main difference between *iPB* modeling principles and that of other data-centric modeling tools [3,4] is that *iPB* does not use workflow notation for describing the flow of work, which is the case with other tools. The order, when needed, is determined by various kinds of business rules.

5 Building and Demonstrating a Model

The project presented in this paper included the following activities (a) three facilitating workshops were all members of the *modeling group* met for brainstorming discussions, (b) modeling work between the workshops based on the discussions and available materials, (c) presentation to and discussion with the *external domain experts*⁵ who did not belong to the modeling group, and (d) writing a report.

The main bulk of the domain knowledge needed for the project came from the modeling group's own experience. Additional knowledge came from a study of the existing systems used for conducting teaching in the department at the time. The traces of the past occurrences of courses in these systems were used to record and discuss scenarios of how these occurrences could look like in the new system.

The first facilitating workshop consisted of informal discussions, the result of which was a general description of the course preparation process presented in Section 2.2 which was made in a *phase-flow* manner. In the second facilitating workshop, the first draft of a *data-centric* process model had been developed, see Fig. 5. This draft was then extended by designing detailed step forms to some steps in Fig. 5; this was done before the third facilitating workshop. The third workshop discussed the first draft by running a scenario of a recently completed course occurrence. Based on this discussion, a list of changes was agreed upon; some of the changes were directly made in the *iPB* model during the workshop. After the workshop, the model was changed according to the list and got the form of Fig. 6. Additionally a relatively full scenario of a past case was recorded using the *iPB* runtime system. Details of the final model are overviewed in Section 6.

After completing the changes, the model was presented to the *external domain experts* (both teachers). The presentation consisted of showing the model in the run-time environment, see Fig. 4 and 3. This was done first by showing how to start preparation of a new course occurrence, and then going through the steps of already recorded scenario. Then the *extended group* (*modeling group + external domain experts*) discussed which requirements on the support system where properly discovered and which were left outside the model and the prototype demonstrated. The conclusions reached in this discussion are overviewed in Section 7.

The results of the project were reported as master thesis written according to the design science research principles.

⁵ For working definitions of terms *modeling group* and *external domain experts* see Section 2.3.

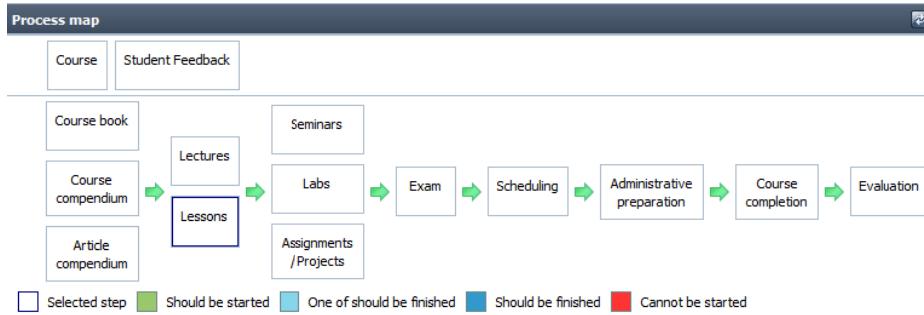


Fig. 5. The first version of the course preparation model

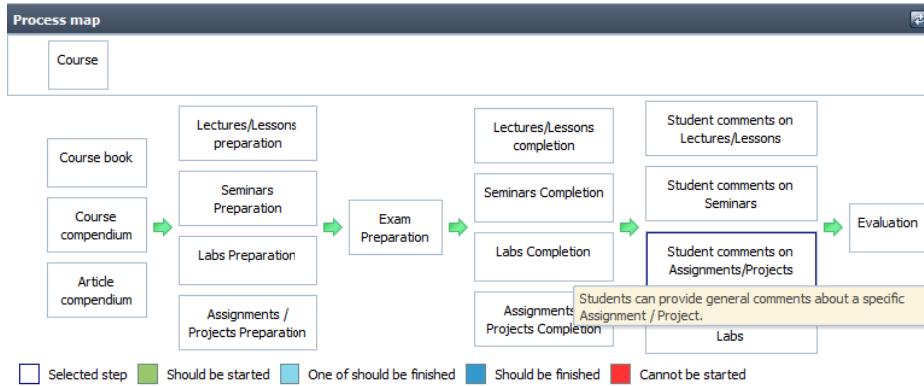


Fig. 6. The final version of the course preparation model

6 The Model Described

The final process map is presented in Fig. 6, Fig. 1 (process map layout in the design studio) and Fig. 4 (an instance process map at runtime). Additionally, Fig. 2 and 3 demonstrate a step form typical for this process, more exactly the form for step "Lectures/Lessons preparation". Fig. 2 gives a step form view in the design studio and Fig. 3 at runtime. As follows from Fig. 6, the process map is clearly divided in two parts: (1) preparation of the course on the left (up to and including *Exam preparation*), and (2) following up the course.

The first part is structured according to the artifacts that need to be prepared for the course, each component being represented as a step in the model (see Fig. 6). This part includes (a) general components: *Course* (description), *Course book* (optional), *Course compendium* (mandatory), *Article compendium* (optional), *Exam preparation*, and (b) specific components that correspond to teaching and learning activities: *Lectures/Lesson preparation*, *Seminar preparation*, *Labs preparation*, and *Assignments/Projects Preparation*.

Each step has a step form attached that defines the data structure for this step. A typical form, the one for step *Lectures/Lesson preparation*, is presented in Fig. 2 (the design studio view), and Fig. 3 (runtime view). It includes the descriptive fields, like: *Title*, *Type*, *Description*, *Lecture presentation*, and the administrative fields, like *Teacher*, *Start*, *Finish*, and *Location*. In addition, it has a journal like widget *Forum for discussing under preparation* where the teachers preparing this teaching/learning activity can leave comments for themselves and for each other. Note that the form in Fig. 3 allows multiple forms at runtime (see tabs) – one for each activity. Multiple forms are also allowed for all other steps except *Course*, *Course compendium*, *Exam preparation* and *Evaluation*.

The system for which the project was discovering requirements was aimed only for course preparation and evaluation; activities for giving the course were left outside as they were more or less covered by the existing systems. The second part of the model, all the steps on the right of step *Exam preparation*, deals exclusively with getting feedback for course evaluation from the teachers and students. More exactly, the steps the names of which end with "completion" are aimed at gathering feedback from the teachers immediately after the corresponding teaching/learning activity. The steps the names of which start with "Student comments on" are aimed at gathering feedback from the student immediately after the corresponding teaching/learning activity. This feedback is aimed to help in step *Evaluation*. An example of a feedback form is presented in Fig. 7. In it, journal field *Students Comments* is designated for recording feedback from students attending the course.

Students Comments		
Date	By	Comment
2012-11-16 08:15	Zakria - Riaz Dar, Zakria	Helpful in resolving any conceptual problems. By Zakria - Riaz Dar, Zakria 2012-11-16 08:08

Fig. 7. Step form for Student comments on Seminars

The feedback steps are synchronized with corresponding preparation steps. When step A is synchronized with step B, the former will automatically get the same number of forms as the latter. If there are references on the A's form to the fields on the B's form, then the references are also synchronized so that they show the contents from the corresponding forms. For example, fields *Seminar title*, *Teacher 1*, *Start* and *Finish* in Fig. 7 are references to the fields on step form *Seminar preparation*, and show their content from the form *Open House Seminar*.

The order of steps in the model is mostly given as a recommended order according to the layout. We found that it is almost impossible to establish a strict order to which all teachers would abide. Though it is highly recommended to have all materials ready before the start of the course, it is happened that changes in some materials and even in the schedule are done very late, when the course has already been started. The business rules where used mainly in its “softest” form – some steps cannot be started unless some other ones has been already started, see their effect at runtime in Fig. 4.

As far as categories of users are concerned, we differentiated two categories: teachers and students. Teachers can access all steps, except that they cannot change any data in the steps designated for gathering feedback from the students. The students will need access to the latter, but not the former except the step *Course* that includes the general information on the course.

7 Analysis of Results and Lessons Learned - Summary

Material presented in this section is based on⁶:

- Own reflections of the *modeling group* (the three authors) over their experience from the project. This is used to answer the *first question* from Section 1, namely, suitability of data-centered process modeling supported by a computerized tool for direct usage in facilitating workshops aimed at discovery requirements.
- Interviews with the two *external domain experts* who were presented the results of our work, but who did not participate in the facilitating workshops. This is used to answer the *second question*, namely, suitability of the approach for presenting the results to the broader audience.
- Protocol of the brainstorming discussion of the *extended group (modeling group + external domain experts)*. This is used to answer the *third question*, namely, how much of requirements could be discovered using a data-centric modeling approach.

Question 1. We came to the positive answer when considering the following *self-reflections*:

- It was relatively easy for us to start modeling in a data-centric fashion. The most important thing to do was to switch the focus from the task flow as described in section 2.2 to the results to be achieved. In our case, the latter was information artifacts to be prepared in the process, compendium, lectures slides, etc.
- Using data-centric approach supported by an appropriate tool inspired our creativity during the facilitating workshops. This was due to highly visual way of representing data-structures as web forms, and possibility of recording past cases. For example, during the third workshop, we discovered that initial presumption that each lecture requires only one teacher and one room does not correspond to the practice accepted in the department. A lecture can be given more than once in the frame of the same course occasion and by different teachers. During the same workshop, we came to the idea of introducing students and teachers feedback gathering during the course, instead of doing it after finishing the course.
- We found it quite convenient to hold discussions on data structures separately from those that concern establishing restrictions on the tasks/operations/activities flow.

⁶ Due to the size limitations, only the summary of results is presented in this paper.

Question 2. We came to the positive answer based on the positive responses from the *external domain experts*. Namely, using *iPB* runtime environment constitutes an efficient way of presenting requirements discovered in the facilitating workshops due to the highly visual means for:

- representing data structures as web forms, see Fig 3.
- representing restrictions on sequence of tasks as grey colored boxes that are ungreyed when situation changes, see Fig. 4.
- recording past cases that were easy to follow by domain experts who were not included in the requirements discovery project.

In addition, the experts appreciated the functioning system prototype provided by the *iPB* runtime environment when it interpreted the process model. It gave quite good understanding of how a process aware system that supported course preparation could look like.

Question 3. During the brainstorming in the *extended group (modeling group + domain experts)* that directly followed the presentation, the consensus was reached that:

- All five types of requirements that we aimed to capture in Section 3.1 were indeed captured in the model to the degree sufficient for starting the system development. To these belong (1) data/information structures, (2) data/information flow, (3) participant collaboration, (4) categories of users, (5) operations included in the process and restrictions on the order in which they can be completed.
- Some requirements that could be of importance were not captured at all. In the first place, this comment concerns requirements on the needs and possible ways of integration with already existing systems. In the second place, this comment concerns the representation of stakeholders' goals. While the first comment is of importance and need to be dealt with in the future, the second one was outside the scope of the project.
- In addition, the open question remains whether an approach taken in the current project can be as good for more complex processes. This comment warrants additional testing.

As far as using *iPB* as a data-centric modeling tool is concerned, our experience shows that it satisfies the criteria listed in section 3.2 sufficiently to be useful in this kind of projects (more detailed analysis of this is not presented due to the size limitations). However, a more visual means for presenting information flow than just field references would be of help in such cases of the tool usage.

8 Concluding Remarks

As was stated in the introduction, the goal of the project was evaluating the suitability of a data-centric business process modeling supported by a tool for requirements discovering. The goal was fulfilled by actually building such a model for a representative process with the help of a tool, and presenting it to the stakeholders. The analysis from Section 7 shows that the approach is valid, but needs further testing and improvement, which will be included in our plans for the future. In particular, new testing would concern using (a) other data-centric process modeling tools, (b) other business processes, and (c) other modeling and domain experts teams.

Our analysis also shows that having a tool that supports scenario recording and prototyping is important. We feel that without these features, the usefulness of the approach we suggest will be limited.

We also believe that our experience report could be of interest for a wider audience than the one that is interested in requirements discovery only. Data-centric process modeling is a relatively new area, and there is not that much experience on its usage reported in the literature. In addition, there is a lack of easily available tools for data-centric business process modeling (see Section 3.2). Therefore, the example and discussions presented in this paper may be of use for any researcher or practitioner interested in non-workflow process modeling. Furthermore, to the best of our knowledge, there is no accepted definition of what data-centric process modeling means and how it differs from other types of process modeling, in particular, artifact-oriented, data-driven, and state-oriented modeling. Discussion on this issue presented in Section 1 may serve as a starting point for clarifying the term and its relationship to other types of modeling.

Acknowledgements. Many thanks to IbisSoft and *iPB* development team – T. Andersson, A. Striy and R Svensson - for providing us with the tool. The authors are also grateful to our colleagues who served as domain experts: M. Henkel and J. Snygg, and to the anonymous reviewers whose comments helped to improve the text.

References

- [1] Alexander, Dukic, L.B.: Discovering Requirements: How to Specify Products and Services. Wiley, UK (2009)
- [2] Bider, I., Johannesson, P., Perjons, E.: Do Workflow-Based Systems Satisfy the Demands of the Agile Enterprise of the Future? In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBP, vol. 132, pp. 59–64. Springer, Heidelberg (2013)
- [3] Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. 32(3), 3–9 (2009)
- [4] Müller, D., Reichert, M., Herbst, J.: Data-driven modeling and coordination of large process structures. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 131–149. Springer, Heidelberg (2007)
- [5] Bider, I.: State-oriented Business Process Modeling: Principles, Theory and Practice. PhD thesis, Royal Institute of Technology, Stockholm (2002)
- [6] Swenson, K.D. (ed.): Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done. Meghan-Kiffer Press, Tampa (2010)
- [7] iPB Reference Manual. On-line documentation,
<http://docs.ibissoft.se/node/3>
- [8] Bider, I., Johannesson, P., Perjons, E.: In Search of the Holy Grail: Integrating social software with BPM. Experience Report. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. LNBP, vol. 50, pp. 1–13. Springer, Heidelberg (2010)
- [9] Jin, X., Wei, H.: Scenario-based comparison and evaluation: Issues of current business process modeling languages. Proc. Inst. Mech. Eng. Pt. B: J. Eng. Manuf. 220(9), 1527–1538 (2006)

A Methodological Framework with Lessons Learned for Introducing Business Process Management

Sebastian Adam, Norman Riegel, and Matthias Koch

Fraunhofer IESE, Fraunhofer Platz 1, 67663 Kaiserslautern

{sebastian.adam,norman.riegel,
matthias.koch}@iese.fraunhofer.de

Abstract. Business process management (BPM) is becoming more and more important for organizations of different sizes. However, the introduction of BPM is a non-trivial task, requiring a lot of experience and helpful guidance in order to be successful. As existing BPM approaches are usually limited to descriptions on a high level of abstraction, they are typically not sufficient to support practitioners in this regard. This paper therefore presents a framework for the introduction of BPM that aims at providing systematic guidance through all the steps of a BPM adoption project. Among other aspects, it especially tackles the areas of a systematic method- and tool selection, which often cause difficulties in practice. In addition, and more importantly, the paper introduces several lessons learned derived from real-world experience made while using this framework. As evidence for the value of these lessons can be presented, they are considered a helpful contribution for industry and academia to make BPM introduction projects more successful.

1 Motivation

Organizations in a competitive environment are facing many challenges, such as globalization, rapidly changing market demands, high expectations with respect to product innovations, flexibility regarding their customers' wishes, and cost pressure. At the same time, they have to fulfill requirements regarding compliance with laws, conservation of resources as well as high rates of return for their shareholders. Since business processes usually form the core of these entrepreneurial actions, the situation described above makes it indispensable for organizations to improve their business processes with respect to efficiency, transparency, and agility [1]. This is the point where business process management (BPM) [2] comes into play. Recent analyses have shown augmented interest in this area [3], and even medium-sized organizations and public organizations such as administrations, research organizations, or universities have therefore started to adopt the BPM concept.

However, while there exist plenty of publications that explain the advantages and principles of BPM, there exists almost no approach describing how to systematically introduce BPM (and supporting toolsets) in an organization with sustainable benefits. Moreover, most existing BPM approaches limit themselves to descriptions on a high

level of abstraction (see related work in section 2) and are therefore hard to apply for less experienced practitioners in industry. Thus, the assumption of many people in the BPM community that it would be implicitly clear how to adopt BPM does not hold true, and we experienced in several projects that such an adoption is still a big challenge. Rather, as this paradigm is quite new, the responsible people in organizations usually do not have sufficient knowledge from their educational background to adopt BPM without clear guidance.

Many organizations therefore need costly external consultancy if they want to introduce BPM, which is often not feasible for small- and medium-sized enterprises or public organizations due to budget restrictions. Such organizations are therefore often prevented from benefitting from the advantages of BPM because the entry barriers are too high for them. This bears the ultimate risk that they may suffer a loss of competitive advantage in the mid-term future.

On the other hand, when organizations try to introduce BPM without the proper knowledge or with inappropriate adoption methods and tools, they risk wasting a lot of effort or even leading the entire organization into chaos. This holds especially true for the selection of BPM methods and corresponding tools to support BPM-related tasks because such methods and tools must be carefully defined to fit the organizational culture. For organizations that are not fully aware of the BPM concepts, a detailed procedure is therefore necessary to support them in selecting suitable methods and tools from the vast quantity of BPM solutions offered. Hence, getting the requirements for a BPM method and a toolset that actually fit an organization's culture is of major importance because both are crucial for the overall success of BPM in practice.

In this paper, we therefore introduce a methodological framework according to which concrete BPM adoption projects can be performed in an organization. The framework takes all the important steps of a BPM introduction into consideration, and resolves the problems mentioned above regarding BPM method definition, tool selection, and requirements analysis. Hence, in contrast to the multitude of rather technical approaches discussed in today's BPM community, our framework concentrates on the methodological aspects of a BPM adoption.

The framework and a concrete instantiating guideline defined by the authors (to be published) have been successfully applied twice in large industry projects so far. By using the lessons learned from the first project in the second, it was possible to significantly increase acceptance and avoid pitfalls. Hence, it can be concluded that using the framework and the annotated lessons learned may help to increase the chances of success in future BPM adoption projects.

Therefore, and as a contribution that is more important than the framework itself, the paper lists and describes these lessons learned for each step of the BPM adoption framework in order to provide a helpful contribution for the industrial audience. However, as the framework and the lessons learned are also usable as requirements for BPM adoption approaches, there is an additional contribution for the academic audience to support them in elaborating more advanced methods in this regard.

The remainder of paper is structured as follows: In section 2, the results of a literature review regarding related work in the domain of BPM introduction are presented. Section 3 introduces the framework for introducing BPM and the research approach according to which it was developed. Section 4 briefly describes two case studies in which this framework was instantiated. In section 5, the lessons learned and

resulting recommendations for BPM adoption projects are explained. Finally, section 6 contains the paper's conclusions and recommended directions for future research.

2 Related Work

A handful of papers and books addressing certain aspects of BPM adoption are already available. In a literature review, we therefore analyze the strengths and weaknesses of the existing approaches, and identify how they are related to the framework introduced in this paper. In this regard, we analyzed whether 1) there is a presentation of the general idea of BPM, 2) a step-by-step guideline for the BPM introduction is included, 3) all aspects of an introduction are covered, and 4) all steps are described in detail. Furthermore, we checked whether they address 5) requirements analysis for a customized BPM method and 6) a BPM toolset. Finally, we investigated 7) how the issue of best BPM practices and recommendations is dealt with. Table 1 summarizes the findings of our literature review.

The BPM handbook “BPM Basics for Dummies” [1] by K. Garimella et al. addresses various aspects of BPM in general and gives an overview of BPM. The strength of the book is the presentation of various BPM best practices and pitfalls, which may enable organizations to introduce BPM more successfully. In addition, it contains an abstract procedure for the introduction of BPM. However, only a small selection of aspects is presented, and neither details nor a concrete step-by-step guidance are provided.

The main part of the whitepaper “BPM Governance” [4] by A.-W. Scheer et al. is the presentation of a BPM lifecycle depicting how to apply BPM in organizations. Additionally, several so-called key elements and best practices are mentioned. Furthermore, the establishment of a “BPM Center of Excellence” is recommended. This institution should support the governance of BPM in an organization and offer various services, such as methodologies, tools, and communication activities. Finally, the paper describes how to establish BPM governance. This part contains all the important steps of BPM introduction; however, it is only very coarse-grained and does not include requirements analysis for a customized BPM method or toolset.

The whitepaper “How to get started with BPM” [5] by Software AG presents the concept of a “process improvement life cycle”, describing several phases from the identification of business processes to the implementation of designed to-be processes. The first steps of BPM introduction include an assessment of the corporate culture, which influences the introduction strategy, and the finding of sponsors to support the introduction. The paper presents the general ideas of BPM and its introduction. Nevertheless, it stays on a high level of abstraction, and neither requirements analysis for the BPM method nor a toolset are touched.

The whitepaper “BPM adoption scenarios” [6] by B. Portier consists of several BPM introduction scenarios, which are presented to extensively describe best practices helping organizations in similar situations. The addressed topics include process modeling and implementation with appropriate tools, the establishment of collaboration among employees, and the permanent managing and monitoring of business processes. However, only general ideas in the form of these scenarios are contained in the paper, while no step-by-step guidance is given. In addition, the paper

mentions some requirements for toolsets, but provides no considerations on how to make an analysis or selection of tools.

The book “BPM Concepts, Languages, Architectures” [2] by M. Weske presents a more detailed methodology for the introduction of BPM. The approach consists of seven phases and covers mostly the important steps that are also included in the framework of this paper. The description of the phases contains general information on what to do, but in some cases it is not specified how exactly to achieve certain results. The selection of tools is included in this approach, but it does not consider a way to systematically analyze requirements for them.

The dissertation “Business process-based Requirements Specification” [7] by J. González contains an approach on business process-based requirements engineering and object-oriented modeling of information systems. Here the aspects of requirements engineering, in particular regarding the elicitation of the as-is situation and the transformation to to-be processes, are described in detail. However, further aspects in the context of BPM introduction are not touched at all or only in passing.

The book “Business Process Management” [8] by J. Jeston et al. contains a complete approach for the introduction of BPM. It consists of ten phases, which cover all the relevant steps in detail, including extensive step-by-step instructions. Best practices are contained as well. Nonetheless, the focus of this approach lies on the development of a proprietary BPM suite, which is why the analysis and selection of existing tools is not included. Additionally, it does not support prioritization and decision-making by providing contain concrete formulas and calculations.

Table 1. Overview of Related Work

Authors	General idea of BPM	Step-by-step guideline for introduction	Complete cover of all aspects	Steps described in detail	Requirements analysis for BPM method	Requirements analysis for BPM toolset	Best practices and lessons learned
Garimella	✓	(✓)					✓
Scheer	✓	✓	✓				(✓)
Software AG	✓	✓					
Portier						(✓)	✓
Weske	✓	✓	✓			(✓)	
González	✓	(✓)		(✓)	✓		
Jeston	✓	✓	✓	✓	(✓)		✓

Thus, as Table 1 shows, there exists some work related to the introduction of BPM in an organization. However, most of these approaches stay on a very general level and do not elaborate the details, and can only be used as an entry point for a company to get a rough insight into the subject of BPM. Especially the whitepapers of BPM vendors are not really useful without additional information or consulting. Despite some very in-depth guidance, such as in the handbook by Jeston mentioned above, the lack of papers or books addressing the topic of systematic BPM tool selection is

remarkable. In addition, the analysis of requirements for customizing a BPM method and selecting an appropriate BPM toolset is hardly considered in the existing literature.

Besides the analyzed resources, a huge amount of work has been done with respect to BPM in general or on particular BPM methods. This is important for an organization to determine whether it makes sense to tackle BPM adoption and invest money in this area at all. However, the introduction of BPM itself is addressed much less intensively and, thus, motivates the need for a systematic framework to support such an endeavor.

3 A Framework for Introducing BPM

As shown in the previous section, according to the authors' experiences made in real projects, no existing BPM approach completely addresses all steps that are important for introducing BPM in a sustainable manner. In this section, we therefore introduce our methodological framework for the stepwise introduction of BPM in an organization. The framework forms a holistic approach in this regard and guides the entire BPM adoption process starting with the initial idea until the final rollout of an organization-specific BPM method and a supporting toolset. By taking into account concrete requirements engineering techniques for capturing the expectations of an organization, the framework particularly aims at guiding the responsible persons through all steps of BPM introduction in a systematic and traceable manner, which is vital for the project's success.

The framework presented in chapter 3.2 is only an abstraction from the concrete method we developed to guide BPM adoption projects. In contrast to known existing approaches in the literature, these guidelines do not only cover step-by-step procedures, but further information such as responsibilities for individual tasks, in- and outputs, or pre- and post-conditions for each activity and are therefore much more detailed. Unfortunately, due to space limitation, only a brief overview of the framework can be provided in this paper.

3.1 Research Approach

The BPM introduction framework was elaborated by using the five steps of “action research” as proposed by Susman [9]. According to this approach, we first identified the actual challenges to be solved when introducing BPM in an organization. These challenges were collected in two real projects (see section 4), where we were asked to support the internal project teams in establishing BPM. The most important challenges we experienced there were

- a) the function-oriented instead of process-oriented thinking of many people
- b) the low trust and high apprehension regarding the advantages of BPM
- c) the unclear implementation of a BPM method in order to actually fit the organizational culture and capacities
- d) the procurement of a suitable BPM suite.

In the subsequent research step of “action planning”, we then collected possible solutions to these challenges. This was done by studying the literature in the

corresponding areas and combining our own experiences from previous projects, as well as our methodological knowledge.

In the third step of “action taking”, we then instantiated and applied the developed framework. We first applied the framework in a medium-sized research institute to gather first experience there. In a second iteration of the action research cycle, we applied an improved version of the framework at a large distributed research campus.

After each execution, an evaluation of the outcomes took place as a fourth research step. By reviewing the work that had been carried out, we analyzed how well the BPM introduction framework fulfilled the demands and addressed the identified challenges. As the researchers developing the framework were directly involved in its execution, they could gain insights into the ongoing activities and judge their success. In the fifth step, the lessons learned were then collected in order to define the required improvements.

The five steps of the action research cycle have been repeated twice so far. The lessons learned described in section 5 were achieved after the second iteration.

3.2 Overview of the Framework

The framework for the introduction of BPM in an organization consists of several steps, whose performance we recommend in order to accomplish BPM introduction projects successfully.

Figure 1 shows an overview of the entire introduction framework and additionally displays the interconnections between the individual steps.

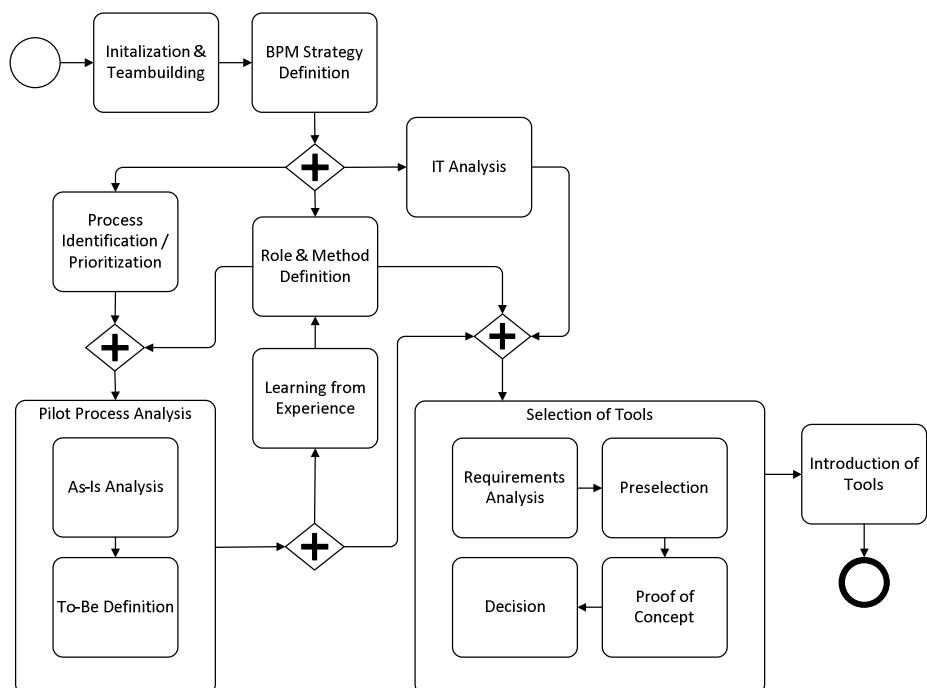


Fig. 1. Steps of BPM introduction framework

The starting point is the “*Initialization and Teambuilding*” step where the idea of introducing BPM is born and the cornerstones for the upcoming activities are laid. The idea can come from any position in the organization. However, to be able to enforce the idea, the essential part is to get the approval of top management. If their commitment is assured, the introduction project is officially launched and an internal team is created that will guide the upcoming steps.

In the “*BPM Strategy Definition*” step, the established team substantiates the initial ideas and analyzes the goals of the organization, the goals to be achieved with BPM, and the goals of the introduction project itself. Furthermore, a strategy regarding how to achieve these goals is defined and corresponding tasks and activities are planned. The strategy must be negotiated with the organization’s top management. Finally, the team derives a measurement plan to allow evaluating the success and progress of the BPM introduction according to the aforementioned goal definition.

The following three steps can then run in parallel. During the “*IT Analysis*” step, the current IT landscape of the organization is analyzed in order to identify which existing systems are to be integrated into the BPM toolset.

During the “*Role and Method Definition*” step, a definition of the roles and method related to BPM takes place. A set of process-spanning and process-specific roles is defined and staffed with real personnel from the organization. Additionally, an organization-specific BPM method has to be defined, which will be applied to manage the organization’s business processes. For some parts of the BPM method, like the as-is analysis and the to-be definition of business processes, the framework already contains a proposal for the corresponding steps. For the remaining parts, references to existing approaches in the literature or tailoring guidance on how to elaborate specific BPM activities for organization-specific needs are provided.

In the “*Process Identification and Prioritization*” step, a decision is made on which business processes will be further investigated as pilot processes during the BPM introduction project, as covering all processes is neither economically feasible nor necessary. Therefore, the existing business processes are captured and prioritized in order to identify those business processes that are suitable for the pilot process analysis.

During the “*Pilot Process Analysis*” step, which starts after process identification and prioritization as well as role and method definition have been completed, these pilot business processes are then worked on. In a first sub-step, they are analyzed to figure out how they are currently being lived in the organization. This is done by performing an elicitation and an analysis of the as-is situation, followed by a specification and validation phase together with representative process participants. In a second sub-step, the processes are then transformed into improved to-be processes by considering the capabilities of modern BPM tools.

When the pilot process analysis is completed for the first selected pilot process, the experiences are collected in the “*Learning from Experience*” step. The aim is to gather all the experiences related to the application of the BPM method in order to use it for adapting the role and method definition towards an approach that fits the organizational context very well. Hence, corresponding retrospective meetings are performed continuously, i.e., after the analysis of each remaining pilot process.

In the “*Selection of Tools*” step, a BPM toolset that fulfills the organizational needs in the best possible way must be chosen. Therefore, the results of the IT analysis, the role and method definition, and the pilot process analysis are used to derive a set of corresponding requirements on the BPM toolset. Based on the derived set of prioritized

requirements, a pre-selection of possible tenderers is then performed in order to reduce the mass of existing offers to a manageable number. Afterwards, proofs-of-concept with the remaining BPM toolsets take place in order to test them extensively regarding their suitability for the organization. Based on the results of the tests, a procurement decision is made.

Finally, in the “*Introduction of Tools*” step, the selected BPM tools are introduced into the organization. This includes not only the installation and integration of the tool, but also the necessary training to enable the employees to use the BPM tools later on. Experiences gained after this step are fed back to the role and method definition if adjustments to the BPM method are necessary. Then, the organization is ready to run BPM in a productive manner.

3.3 Instantiating Method

As already mentioned above, we have defined a concrete, instantiating method for managing BPM adoption projects (still to be published) besides the generic

Table 2. Example for the description of a step of the introduction method

Name	Purchasing decision
Goal	Making the decision to buy a BPM suite.
Precondition	Show case is completed.
Input	Ranking of BPM suites.
Involved Stakeholders	Members of the total team.
Procedure	<p>1. Calculate the cost-benefit ratio (CBR) for each of the tested BPM suites. To do so, divide the score of the BPM suite determined after the show case (compare chapter 3.9.3) by its price, which was elicited during the self-disclosure.</p> $CBR_{Tenderer} = \frac{Score_{Tenderer}}{Cost_{Tenderer}}$ <p>To normalize this value, divide it by the highest CBR value in order to get a value between zero and one for each tenderer.</p> $CBR'_{Tenderer} = \frac{CBR_{Tenderer}}{\max \{CBR_{Tenderer}\}}$ <p>2. Rate the tenderer's experience and stability (EaS) on a scale from one for very poor to five for very good. To do so, make use of information about the company such as its economic situation, the tenderer's references and experiences. This information has been gathered during the self-disclosure. To normalize the value, divide it by 5.</p> $EaS_{Tenderer} = \frac{RatingOfEaS}{5}$ <p>3. ...</p>
Output	Recommendation to buy one particular BPM suite.
Post-condition	Decision to get one particular BPM suite is made.

methodological framework. This method is designed to give a prescriptive guideline in order to enable practitioners to run BPM adoption projects in a repeatable manner. For this purpose, detailed information on what has to be done when, how, and by whom, including decision procedures, moderation guidelines for workshops, formulas for various calculations, inputs, outputs, pre- and post-conditions for activities, etc. are provided. An example of how the steps are described is depicted in Table 2.

4 Real-World Application Studies

The framework, respectively the aforementioned instantiation, has been applied in two large projects so far. The context of these projects is briefly depicted below.

4.1 Engineering Institute

The first project started in late 2010 and took place in an institute dealing with engineering consulting. The institute is organized into eight departments with around 250 employees in total. The overall motivation of the institute for introducing BPM was the low effectiveness and efficiency of supporting business processes, as this resulted in low process compliance and dissatisfaction among the employees. In particular, too much effort was spent on administrative or supporting issues, which negatively affected the results of the core processes. After motivating top management to think about BPM, a BPM core team was established in the organization involving administrative people as well as external experts (i.e., two of this paper's authors). Together with the management, this team defined precise goals for the BPM introduction and derived a clear strategy for what the organization wanted to achieve by using this paradigm. Based on this, the core team then started involving further stakeholders from other departments such as the head of the administrative department or administrators from the IT department. With the former person, an initial business process map was created using brainstorming techniques before systematic prioritization took place in order to determine which processes should act as pilot processes. With the IT administrators, the constraints given by the IT landscape (e.g., hardware, backup systems, firewalls, LDAP, etc.) as well as the most important information systems to be potentially integrated into a BPM suite were then identified. In parallel, the core team tailored the BPM method and the role model proposed by the authors to the given situation at the institute. Most importantly, concrete persons within the institute were assigned to the cross-cutting and process-specific roles prescribed by the framework in order to determine the stakeholders to be involved in the next BPM adoption steps.

Together with the people assuming a process-specific role in the (three) identified pilot processes, these processes were then analyzed in detail. Both the as-is and the to-be analysis took place in joint workshops lasting two hours each. In between, the core team elaborated the information gathered, specified the pilot processes in their as-is respectively to-be state, and conducted validation interviews with the stakeholders. After each workshop, a retrospective meeting was held in which the procedure for the next workshop was defined based on the lessons learned.

Based on the results of the pilot process analysis and the consideration of IT constraints, an initial set of requirements for a BPM suite to be procured was derived by the core team. Furthermore, the core team elicited additional requirements from people who had been assigned to the cross-cutting BPM roles before. The purpose of this additional elicitation was to understand their functional and non-functional expectations concerning software support for generic BPM activities like process modeling, process implementation, process controlling, etc.

The final list of requirements was then prioritized and used to contact potential BPM suite vendors. From nine contacted organizations, five organizations were preselected and invited to present their solution and to explain how they would deal with the stated requirements. Based on the impressions gathered during these presentations, a procurement recommendation had to be made to the top management because no further effort was allocated for making a more in-depth proof-of-concept. Unfortunately, due to political reasons, it was not the recommended solution that was finally bought at the end of 2011. In the meantime, and with some corresponding challenges, the pilot processes have been implemented and are currently going to be brought to production.

4.2 Large Basic Research Organization

The second project started in the middle of 2011 and was carried out at one of the largest organizations for basic research in Germany. The institution employs more than 2000 people organized in more than 60 departments distributed over a large distributed campus. In addition, the organization accommodates and involves several thousand guest researchers per year in its core research and development activities. The overall motivation of this institution was to establish a process-oriented culture in order to perform the high number of internal processes much more efficiently and with much higher transparency for all involved people. After getting commitment from the organization's top management, a core team staffed by people from the internal process department and external experts (the paper's authors) was created. According to the proposed framework, the core team then started to identify and select business processes to be treated as pilot processes during the adoption project. Important criteria were quick wins when implementing these processes, representativeness for the organization's processes in general, and a crossover nature in order to be able to involve people from different departments. In parallel, the 15 core IT systems were identified and analyzed in terms of their suitability for integration or even replacement by an integrated BPM suite. Furthermore, the core team defined an initial BPM method to be used for the pilot process analysis and identified and involved people acting as process-specific roles for the pilot processes.

Unfortunately, it was not possible to determine people to take over the cross-cutting roles: Top management was not willing to set up a BPM organization as long as no decision basis had been provided yet by means of a procurement recommendation. Together with the involved people, a pilot process analysis then took place; again separated into an as-is and a to-be analysis phase. However, in contrast to the first project, much more effort was needed here. So, for each pilot process, up to three workshops were needed for the as-is, respectively to-be phase. The reason was that

there was a multitude of department-specific variants, which had to be aligned and negotiated. Thus, several lessons were learned that required the core team to adapt the analysis procedure continuously. Based on the identified characteristics of the pilot processes in their indented to-be situation as well as on the results of the IT analysis, requirements for a BPM suite were then derived. However, as no people were assigned to the cross-cutting roles, e.g., for process modeling or implementation, the core team had to derive requirements from these perspectives on their own. In this context, the reuse of requirements specifications from the first project was helpful to complement the initial set of requirements by adding BPM-task-specific issues. Instead of contacting BPM vendors directly, an official request for proposals was made by the organization. After making an initial pre-selection based on hard K.O.-criteria, 13 remaining vendors were invited to give a brief demonstration. Based on further criteria checked during these demonstrations, only three vendors made it to the final proof-of-concept, in which dozens of real-world scenarios were played in order to test the solutions for their suitability in the organization. Thus, a well-founded procurement decision could be made at the end of 2012.

5 Lessons Learned and Recommendations

The lessons described in this section were learned after applying the framework in the two aforementioned BPM introduction projects. We publish the lessons learned for two reasons:

1. To invite other researchers to contribute their ideas to continuous improvement of this framework.
2. To inform practitioners dealing with BPM adoption for the first time about good practices and typical pitfalls.

In the following, we describe our experiences for each step of our method that we deem as most relevant.

Initialization & Teambuilding: In this step, we experienced that it is very important to get commitment from management at higher levels (preferably top management) for the BPM project. BPM is a strategic issue, which includes cultural changes in the whole organization. Effectiveness can only be achieved if processes are analyzed across the boundaries of organizational divisions, which can often only be enforced by management decisions on high levels. In building the core team for the BPM adoption, it is important to choose team members who are technically versed and personally appreciated across disciplines and by various hierarchy levels. The core team should be able to motivate, influence, and fascinate others to push the project forward. This will ultimately increase acceptance for BPM in the whole organization. In addition, it is important to release the core team members from other work, so they can focus on the BPM introduction. Otherwise, the BPM project might easily lose priority while competing with the ordinary daily tasks of the core team members.

BPM Strategy Definition: It is a central aim to clearly determine why an organization is willing to invest in BPM and what BPM is supposed to provide. Thus, the goals to be achieved by BPM but also by the BPM adoption project itself should be defined precisely. Thus, clear controlling and measurement plans should be derived early in order to be able to track progress and success continuously. However, it is not

only necessary to define a clear BPM strategy but also to communicate it in the organization in order to avoid wrong hopes or fears (e.g., fear of people of losing their jobs) that might arise. Ideally, employees should be involved directly in decision workshops, because BPM will not work if it is enforced in a patronizing manner.

IT Analysis: In this step, the analysis should only focus on important parts of the IT landscape regarding a BPMS to be procured. Otherwise, too much effort will be spent on the analysis of systems of minor importance, which will consume a lot of time and effort. The IT analysis should therefore primarily analyze the existing infrastructure systems such as LDAP, etc., the existing business applications with which the daily tasks are performed, and any further system that is already known to be replaced or integrated by a BPMS.

Process Identification & Prioritization: We experienced that it is important to select representative processes with quick wins that are not too complex. Considering complex processes during BPM adoption will lead to getting stuck in the process analysis without seeing results. This might lead to negative effects like disappointment among stakeholders or budget overruns already during the pilot phase, which will cast a shadow on the whole BPM initiative. On the other hand, addressing processes with low representativeness for an organization's process landscape or with low value is also risky, as such analyses would probably lead to wrong conclusions regarding both the required BPMS functionality and the overall improvement potential of BPM.

Role & Method Definition: As in core team building, it is also relevant for the actual BPM role staffing to achieve releases for these roles and make BPM one of their daily tasks (or ideally the only one). In this regard, it is essential to consider existing skills and competencies regarding business process modeling, controlling, implementation, etc. Furthermore, it is important to adapt the overall BPM concept to the actual needs of the organization, as every organization has different requirements regarding certain aspects (e.g., business process modeling language, elicitation techniques, etc.). In this regard, it is essential to ensure that a BPM method fits the organizational work culture. For instances, in highly flexible organizations with low standardization, it is very risky to roll out an approach that tries to rigidly govern processes. Thus, establishing a BPM method without systematic tailoring based on a thorough analysis of the actual organizational needs is a risky undertaking.

Pilot Process Analysis: Regarding the participants to be involved in process analysis workshops, we experienced that it is important to make it transparent who is chosen for the workshop and why. If possible, other stakeholders may also be informed about why they were not chosen. Representatives for all relevant roles of a business process (at least process participants and process owners) have to be included in order to get a complete picture of the process. For this purpose, we recommend also involving superiors as they often have an overview of large parts of the process. Furthermore, confidentiality should be claimed and assured at the beginning of each workshop. It must also be made clear for the participants that not people and their work are being criticized, but only the processes in their current state. These social aspects should lead to an open atmosphere, because otherwise information might be repressed, leading to wrong process descriptions and requirements. Regarding the performance of the as-is analysis workshop, it is not useful to model the processes directly electronically, e.g., using a laptop and presenter, because this takes too much time and detracts people from discussing the as-is situation. It is also not recommended using a process

modeling notation during the workshop because it is quite likely that many of the involved people are not able to understand it, even though they will probably not complain about that. We therefore recommend using meta-plan methods and cards, as these means involve the participants more actively in the creation of the process description (a formal language can be used afterwards to document the process in the back office). In this regard, it should also be focused on the main flow of the (as-is) process, thereby avoiding getting lost in details, exceptions, and branches. We recommend regularly reminding the participants that the 80% case of the process should be elicited. It is reasonable to identify the exceptions but not to model them in the process at this point in time. However, eliciting the perceived strengths and weaknesses of the as-is processes from the participants is an important step that should not be neglected.

Regarding the to-be process definition, we made good experiences by creating a proposal offline on which the participants can comment in a joint walkthrough. It is important to link the identified weaknesses to changes incorporated in the process. This makes it clear how these weaknesses will be addressed in the new process, which gives the participants a better understanding of what BPM and especially a BPMS is supposed to do. However, the recommended or possible process changes should be prioritized, as often not all aspects might be realizable within a given time and budget. Thus, decision makers should be involved early on. Furthermore, during the to-be definition, it is also necessary to explicitly and implicitly define the given exceptions as well as clear business rules. In particular, we recommend carefully analyzing the given business rules (and especially their enforcement), as overly strict rules may hamper an efficient flow of the process (or even block it), which in turn might lead to several problems in daily business such as a low acceptance of a BPMS, or uncontrolled workarounds.

Learning from Experience: In order to have an effective retrospective meeting we recommend collecting feedback and writing down the experiences continuously, at least after each workshop during the pilot process analysis. However, it is not sufficient to just track these lessons – the method must also be adapted accordingly. The core team should therefore analyze possible alternatives regarding how to avoid an observed problem and discuss which solution would be most promising. This is a good basis for continuously optimizing the method definition for the organizational context.

Selection of Tools: In order to filter the high number of possible BPMS tenderers, it is essential not just to state clear requirements, but also to define hard K.O. criteria. Furthermore, we recommend only inviting tenderers to further presentations that can fulfill at least 60% of the stated requirements. The presentations should have a strict agenda and timeframe. This prevents lengthy presentations about information of minor importance (e.g., pure marketing presentations). After that, the number of tenderers should have already been reduced to two to four tenderers, which will then be invited to an in-depth proof-of-concept workshop. In preparation for that event, we recommend elaborating concrete test cases that are comparable to the daily tasks of the organization when using the BPMS later. We propose including challenging scenarios to ensure that the tools are tested appropriately in the areas of process modeling, process implementation, process execution, and process monitoring. However, in the final decision, the economic situation of the tenderer should also be regarded, as well as the size and turnover of the company in comparison to the estimated project size.

Table 3. Relevance of lessons learned

Step	Lesson learned	Relevance
Initialization and Teambuilding	Get commitment from management at higher levels	Medium
	Choose technically versed and personally appreciated members	Medium
	Release the core team members from other work	High
BPM Strategy Definition	Determine clearly why the organization invests in BPM and what BPM is supposed to provide	Medium
	Derive clear controlling and measurement plans	Low
	Involve employees directly	High
IT Analysis	Focus only on important parts of the IT landscape	Medium
Process Identification /Prioritization	Select representative processes with quick wins	High
Role & Method Definition	Achieve releases for people taking BPM roles	High
	Consider existing skills and competencies	Medium
	Adapt the BPM concept to the needs of the organization	High
Pilot Process Analysis	Make transparent who is (not) chosen for the workshops	Medium
	Involve all affected process roles (including superiors)	High
	Claim confidentiality	Low
	Use meta-plan methods to model the process instead of doing it by using a laptop and modeling languages	High
	Focus on the elicitation of the main flow of the process	High
	Elicit perceived strengths and weaknesses of the process	Medium
	Create a proposal for the to-be process offline and perform a joint walkthrough over the process with the participants	High
	Link the identified weaknesses to changes in the process	Low
	Prioritize possible process changes	Medium
	Involve decision makers early on	High
	Analyze given business rules carefully	Medium
	Collect feedback and experiences continuously	Low
Learning from Experience	Adapt the BPM method continuously based on experience	High
Selection of Tools	Define hard K.O. criteria to filter tenderers early	High
	Only invite tenderers to presentations that can fulfill at least 60% of the requirements	Medium
	Have a strict agenda and time frame for the presentations	High
	Select only up to four tenderers for a proof-of-concept	Medium
	Elaborate test cases that are comparable to daily tasks and include challenging scenarios	High
	Consider the economic situation of the tenderer	Medium
Introduction of Tools	Share team members' knowledge with their successors	Medium
	Define a clear rollout strategy and a strategy for the time after the BPM adoption project	High

Here, it should be checked whether the tenderer is really able to provide a solution that is not only appropriate technically, but also with regard to service levels, for example.

Introduction of Tools: If members of the core team will be substituted by other personnel after the adoption project, it is important for them to share their knowledge

with their successors promptly to ensure fluent transition between the introduction and future BPM activities. Furthermore, a clear rollout strategy (e.g., installing, configuring, training, etc.) must be defined and followed; as otherwise, the project might get stuck close to the end. Especially when the core team is released from its responsibilities, it must be clear who will further drive and manage the BPM activities. Without a motivated, skilled, and acknowledged person to do this job, the aforementioned investments are jeopardized. Thus, it is essential to define a clear strategy for the time after the BPM adoption project.

In Table 3, we summarize our lessons learned and assess their relevance based on practical experience from the aforementioned case studies and further projects. High relevance means that we run into problems when not following a lesson while we did not run into these problems when following it. Medium relevance means that when applying a lesson we experienced success, while we made no experience when not following it. Low relevance means we did not notice any difference, no matter whether we applied a lesson or not.

6 Conclusion and Future Research

An increasing number of organizations are currently interested in adopting business process management (BPM) to gain a competitive advantage by exploiting the benefits of this paradigm.

However, while there are plenty of publications dealing with BPM aspects in the area of notations, technologies, activities, or governance / management issues, methods or guidelines that explain how to introduce projects with sustainable results are rare, or described only on a very high level of abstraction. The majority of organizations have no staff with solid BPM experience, which represents a high entry barrier, as it bears the risk of running into problematic pitfalls if no investment is made in external consultancy.

Existing approaches like the handbook of Jeston [8] already cover to some extent the steps of BPM adoption as proposed in our framework. However, in some crucial aspects we made different experiences in practice, and therefore include them in our framework. For example, in contrast to using a combination of laptop and video projector for business process elicitation as suggested by Jeston, we experience the achievement of much better results by using the (informal) meta-plan method, as indicated in the lessons learned.

The purpose of this paper is therefore the provision of a methodological framework (based on practical experience) for guiding practitioners through the typical and most important steps of BPM adoption. Thus, for each of these steps, we presented our lessons learned gathered in two large real-world projects in order to provide practitioners with some insights when running their own adoption projects. Thus, it is expected that typical pitfalls can be avoided and that greater project success becomes realistic. However, the lessons learned also indicate areas where future research and development are worthwhile. Thus, our paper aims at providing first ideas for researchers based on our experience in this regard.

For the future, empirical studies are planned in order to provide more evidence of the advantages of our methodological differences in comparison to other approaches.

References

- [1] Garimella, K., Lees, M., Williams, B.: BPM Basics for Dummies. Wiley Publishing (2008)
- [2] Weske, M.: Business Process Management Concepts, Languages, Architectures (2007)
- [3] Casewise, The Business Process Management Marketplace by Numbers (September 2011), <http://www.casewise.com/docs/products-downloads/casewise-bpm-infographic.pdf> (accessed February 08, 2013)
- [4] Scheer, A.-W., Brabaender, E.: BPM Governance The Process of Business Process Management (2011)
- [5] Software AG, How to Get Started with BPM: Ramping Up Process Initiatives & Overcoming Hurdles to Success (2012)
- [6] Portier, B.: Business process management adoption scenarios (2011)
- [7] de la Vara González, J.L.: Business process-based Requirements Specification and Object-oriented conceptual Modelling of Information Systems (2011)
- [8] Jeston, J., Nelis, J.: Business Process Management Practical Guidelines to Successful Implementations (2006)
- [9] Susman, G.I.: Action Research: A Sociotechnical Systems Perspective (1983)

Repairing Business Process Models as Retrieved from Source Code

María Fernández-Ropero^{1,2}, Hajo A. Reijers^{1,3}, Ricardo Pérez-Castillo²,
and Mario Piattini²

¹ Department of Mathematics and Computer Science, Eindhoven University of Technology
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands

{m.fernandezropero,h.a.reijers}@tue.nl

² Instituto de Tecnologías y Sistemas de la Información, University of Castilla-La Mancha
Paseo de la Universidad 4, 13071 Ciudad Real, Spain

³ Perceptive Software

Piet Joubertstraat 4, 7315 AV Apeldoorn, The Netherlands
{marias.fernandez,ricardo.pdelcastillo,mario.piattini}@uclm.es

Abstract. The static analysis of source code has become a feasible solution to obtain underlying business process models from existing information systems. Due to the fact that not all information can be automatically derived from source code (e.g., consider manual activities), such business process models may not entirely reflect the original behavior of the organization. This paper provides a technique to repair such business process models on the basis of event logs collected during the execution of information systems. The technique detects missing sequence flows regarding the event log and tidily adds these sequence flows to the target business process model. In order to enhance its applicability, this technique is tool-supported. Additionally, this paper provides a case study with a real-life system to demonstrate its feasibility.

Keywords: process models, source code mining, event logs, repairing.

1 Introduction

Business process management enables organizations to become more efficient, more effective and more readily adaptable to changes than traditional, functional management approaches. Business processes describe the organization's operations, as well as the roles and resources involved [1]. Sometimes, however, business processes models do not explicitly exist in an organization. And even if an organization has created models of its business processes, these could be outdated and misaligned with the actual activities. In cases where business activities are supported by information systems, reverse engineering techniques can be used to obtain business process models from these. This is often an attractive practice, since existing information systems may embed business logic in their source code. For this reason, *business process archaeology* has emerged as a set of techniques and tools to mine business processes from source code. Source code contains a lot of business knowledge that has been

embedded during the information system maintenance. Thus, business process archeology represents a good start point for business experts, requiring less effort than modeling from scratch. One of these techniques is MARBLE [2], which is based on a model-driven approach, and uses the KDM (Knowledge Discovery Metamodel) standard to represent intermediate models.

While the analysis of source code allows the acquisition of embedded knowledge that is not present anywhere else, their application may entail a semantic loss due to the increase of abstraction level [3]. Business process models obtained in this way can therefore be incomplete, could contain irrelevant information, or may even contain ambiguities that decrease their understandability. The improvement of such a processes model is necessary to address these problems, which helps to have them better reflect reality [4]. To enrich the semantics of business process models it is necessary to consider alternative sources from which to extract knowledge. *Event logs* form one such candidate. In an opposite way to business process archeology, process mining techniques aim at obtaining useful information from event logs by means of process discovery, conformance checking and model enhancement [5, 6]. These event logs are recorded by information system such as enterprise resource planning (ERP) or customer relationship management (CRM) systems, among others, i.e., process-aware information systems (PAISs) [7]. Organizations may also operate traditional (non-process-aware) information systems supporting their business processes, which do not record any event during execution.

This paper presents a technique to repair business processes models as obtained by a static analysis of source code by capturing additional information from event logs. To develop the technique two assumptions based on our previous work were taken into account: (1) business process models, capturing a static viewpoint, are obtained by means of MARBLE, an adaptive framework to recover the underlying business process models from legacy information system; and (2) event logs, representing a dynamic viewpoint, are obtained by means of the technique proposed by Pérez-Castillo *et al.* [8] in which event logs are generated from non process-aware systems, which enables a process mining approach. Business process models obtained with these two different techniques display similarities as well as differences. Hence, our proposed approach finds similar tasks in both models in order to detect missing sequence flows by comparing both artifacts, i.e. those sequences flows that can be inferred from the event log but which are not in the initial business process model. The detected, missing sequence flows are incorporated into the target business process model, making it more complete and accurate regarding to the event log. The actual improvement obtained after this repair step is evaluated in the paper through a case study using a real-life information system. The case study's results show that the repaired business models are indeed more accurate and more complete than the initial model as retrieved by reverse engineering.

The remainder of this paper is organized as follows: Section 2 presents related work. Section 3 introduces the proposed approach to repair business processes models using event logs. Section 4 shows some preliminary results provided by the proposed approach using real-life systems. Finally, Section 5 presents the conclusions and directions for future work.

2 Related Work

In the literature, various techniques are described to obtain business process models. Some of these techniques consider dynamic analysis, which obtain process models from the event logs that are recorded during system execution. These logs represent the actual system performance and several algorithms can be used, such as the alpha algorithm proposed by *Van der Aalst et al.*, a genetic algorithm proposed by *De Medeiros et al.*, a heuristics algorithm proposed by *Weijters et al.*, among others, to mine the business process [9-11]. The event logs used by these algorithms are obtained from process-aware information systems (PAISs) [7], i.e., information systems whose nature facilitates the direct registration of events throughout process execution. Although information systems that are not process-aware do not automatically record event logs, such logs can be obtained by hand or by injecting code to trace by techniques as proposed by *Perez-Castillo et al.* [8]. These event logs are generated when the information system is running, and describe which tasks are executed and in what order for a certain time period. The downside of such event logs is that not all functionalities can be captured, i.e. only tasks that have been carried out at the time of executing the injector. That is, if the injector stores the event logs for a year it is not possible to recover the tasks that are executed, e. g., two years back, or it may not be able to recover those tasks that hardly ever occur but are important for the system.

Apart from dynamic analyses, a static analysis has been proposed. Static analysis obtain process models through the syntactical analysis of the source code, e.g. by *Zou et al.* [12]. They developed a framework based on a set of heuristic rules to extract business processes following model-driven development. The framework statically analyzes the legacy source code and applies the rules to transform pieces of source code in business process elements. Although this work is based on the MDA (Model-Driven Architecture) approach, standards as KDM are not considered. *Ghose et al.*[13], in turn, consider other software artifacts as a set of text-based queries in documentation for extracting business knowledge, but the approach is not based on the MDA. *Perez-Castillo et al.*[2], use standards in their approach to obtain process models. They propose MARBLE to obtain a first approximation of business process that is especially useful for organizations that have never modeled their processes, while their legacy information systems do embed knowledge during its maintenance (knowledge that is only present in the source code, not in the documentation). Unfortunately, the retrieved process models have a low abstraction level, being very close to the code level. Furthermore, not all embedded information can be obtained using MARBLE. Thus, the recovered process models involve several challenges to address.

Neither static nor dynamic analysis can obtain the actual and complete contours of business processes in an organization. *Adriansyah et al.* [14] discuss in their work that a retrieved model often does not describe the process executions as observed in reality, e.g., activities in the model are skipped in the log, the log contains events not described in the model or the order execution of the tasks are different. This work compares the process model with an event log of the same process. In follow up to this observation, *Fahland et al.* [4] suggest to repair business process with the recorded event logs. They obtain subprocesses in event logs not being present in the process model and then, insert them where it is missing. This particular work assumes that the process model has been discovered by mining process (using event logs) or by hand.

However, this is mostly realistic in PAIS settings. The present paper is focused on a technique to repair business process using event logs that are also suitable for non-PAISs. Thus, this work combines the static and dynamic analysis in order to improve process models.

3 Technique for Repairing Business Process Models

The repair technique combines artifacts obtained from the static and dynamic analyses of existing information systems, i.e., a first sketch of business process models and event logs collected at runtime. The main goal of the technique is to detect missing sequence flows by comparing both artifacts and build an improved business process model containing these sequence flows. The technique has been defined under two assumptions, which are related to the two previously mentioned techniques. Despite these two assumptions, this approach can be adapted to other techniques with which to reverse business process models or obtain event logs.

Assumption 1. One of the assumptions of the repair technique is that the process models are obtained using MARBLE (Modernization Approach for Recovering Business process from LEgacy systems) [2], a framework for obtaining business processes from legacy information systems (LIS for short), focusing on the phase of reverse engineering. MARBLE is based on KDM, which is recognized as an ISO/IEC 19506 standard [15] and allows abstract conceptual representations of the different views of the architecture of legacy information systems. Afterwards, this knowledge is gradually transformed and refined down to the underlying business processes. For this purpose, MARBLE is divided into four levels of abstraction and defines three transformations. In order to achieve optimal business process management, MARBLE represents business processes by means of Business Processes Model and Notation (BPMN) [16]. This notation is a well-known graphical notation and is aimed to be easily understandable by system analysts as well as business analysts.

Assumption 2. The second assumption is that event logs are obtained by the injection of fragments in specific parts of the information system to generate an event log file during system execution, using the event traces injector proposed in [8]. This approach generates event logs in MXML (Mining XML) format from non-process-aware information systems. Although the technique is generic, the supporting tool that is used in this work, Event Traces Injector (ETI), has been designed for object-oriented systems. Event logs are considered as a suitable knowledge source to discover what is really going on in an organization. Each event log is related to a “run” of the process, i.e., a process instance, and provides additional information about the resource executing or initiating the activity, the timestamp, or data elements. Process mining [17] aims at knowledge extraction from event logs available in information systems. Among the available process mining techniques, this paper uses the Heuristic Miner algorithm. The Heuristics Miner proposed by Weijters *et al.* [11] uses a heuristic approach to provide the control flow of the information system from an event log. It is usually applied to real-life data with not too many different events, or for carrying out further analysis in PROM [18].

Fig. 1 shows the sequence of steps carried out to obtain an improved process model ('Process model'). The start points of the technique are the process model and the event logs. The steps are described in Sections 3.1 to 3.4. To facilitate their understanding, a running example will be progressively developed in the mentioned sections. They relate to a real-life information system, in which the technique is applied to Villasante-Lab, a company devoted to the chemical analysis of water and waste water (cf. Section 4).

3.1 Step 1: Obtain Info Tasks and Diagrams

This step analyzes, on the one hand, the business process according the BPMN notation and, on the other hand, event logs according to the MXML notation. In the process model, each diagram (BusinessProcessDiagram) contains several tasks, data objects, and inter-connections between these. In event logs, the name of each event corresponds to the name of the class to which it belongs and the name of the method invoked (nameClass.nameMethod). The nameMethod is considered the task name, while the nameClass is considered the diagram name in which the task is contained.

This step obtains which task is included in which diagram. Diagrams are classified as fine-grained or coarse-grained diagrams in order to apply different treatments depending on the type of granularity (e.g., in an object-oriented system, MARBLE transforms some classes in BPMN diagrams and other as tasks inside another diagram while ETI considers each class as a diagram). This classification is made according to a proposed limit. This signifies that if a diagram contains fewer elements than this limit specified as the number of tasks, then that diagram is considered as a fine-grained diagram.

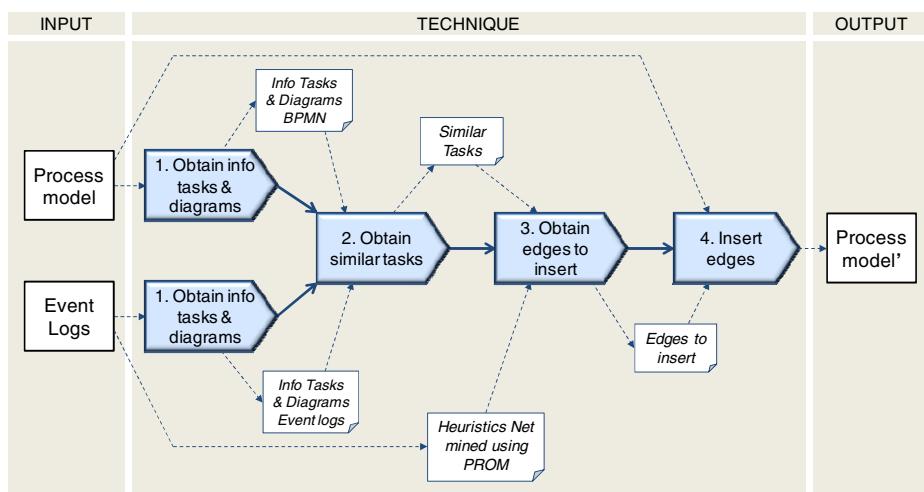


Fig. 1. Technique to repair BPMN using Event logs

To continue with the running example, Table 1 shows the diagrams obtained after applying both techniques to Villasante-Lab and the number of tasks in each of these. As the tables show, most MXML diagrams are fine-grained and contain very few tasks (usually one) while the BPMN part contains less fine-grained diagrams. Thus, some MXML diagrams correspond (or are equivalent) to tasks in BPMN diagrams. In this running example, the limit to characterize fine-grained diagrams is one task. Besides, a task may contain several occurrences in different diagrams as MXML tasks `getUserManager` and `setInvoiceManager` in Table 1.

3.2 Step 2: Obtain Similar Tasks

The information mined from information system using both techniques (MARBLE and Event Traces Injector) displays the following differences:

- **Different types of granularity.** Depending on the extraction techniques, the diagrams show different types of granularity, e.g. in an object-oriented system, MARBLE considers some classes as BPMN diagrams, while other classes are considered tasks inside another diagram, whereas ETI considers classes as diagrams.

Table 1. Extract of tasks Information. BPMN.

	Name task	Name diagram	Type	Diagram
BPMN	BaseZoneController	AddPointAdminController	Coarse	
	getUserManager	BaseUserController	Coarse	
	getClientManager	BaseClientController	Coarse	
	initBinder	AnalysisBean	Coarse	
	doPrepareView	AnalysisBean	Coarse	
	searchZoneNoHistoricas	AnalysisDAO	Coarse	
	searchZone	AnalysisDAO	Coarse	
	filterUser	AnalysisDAO	Coarse	
	convertDissolutionToDissolutionBean	BaseDissolutionController	Coarse	
	calculateTotal	BaseDissolutionController	Coarse	
MXML	Transform	PdfExport	Fine	
	resolveException	ExceptionResolve	Fine	
	setZoneManager	BaseZoneController	Fine	
	getUserManager	AuthenticationManager	Coarse	
	getUserManager	BaseUserController	Coarse	
	setRolManager	BaseRolController	Fine	
	setInvoiceManager	BaseInvoiceController	Fine	
	setInvoiceManager	BaseLinesInvoiceController	Fine	
	setDissolutionManager	BaseDissolutionController	Fine	
	getClientManager	BaseClientController	Coarse	

- **Not Covering the Same Number of Tasks.** While the BPMN model contains all the business tasks derived from source code, the MXML model only contains those tasks executed during the ETI execution during a certain time. The executed tasks outside the execution period are not recorded and neither are those tasks that rarely occur but are important for the system. Following with the running example, Villa-sante-Lab, 368 business tasks have been obtained in the BPMN model while only 96 tasks appeared in the MXML model. This represents that the execution of this information system during that time only executed 26% of business tasks of the whole instrumented information system.
- **Similar Tasks.** The tasks used in these two models also display similarities. The great challenge is to know which tasks of the MXML model correspond to tasks of the BPMN model (see Fig. 2). This is done by computing the syntactic distance of their name labels. When a MXML task is contained in a fine-grained process, it is necessary to compare the MXML diagram with each BPMN task (due to different granularities) as well as to compare the names of the MXML and BPMN tasks.

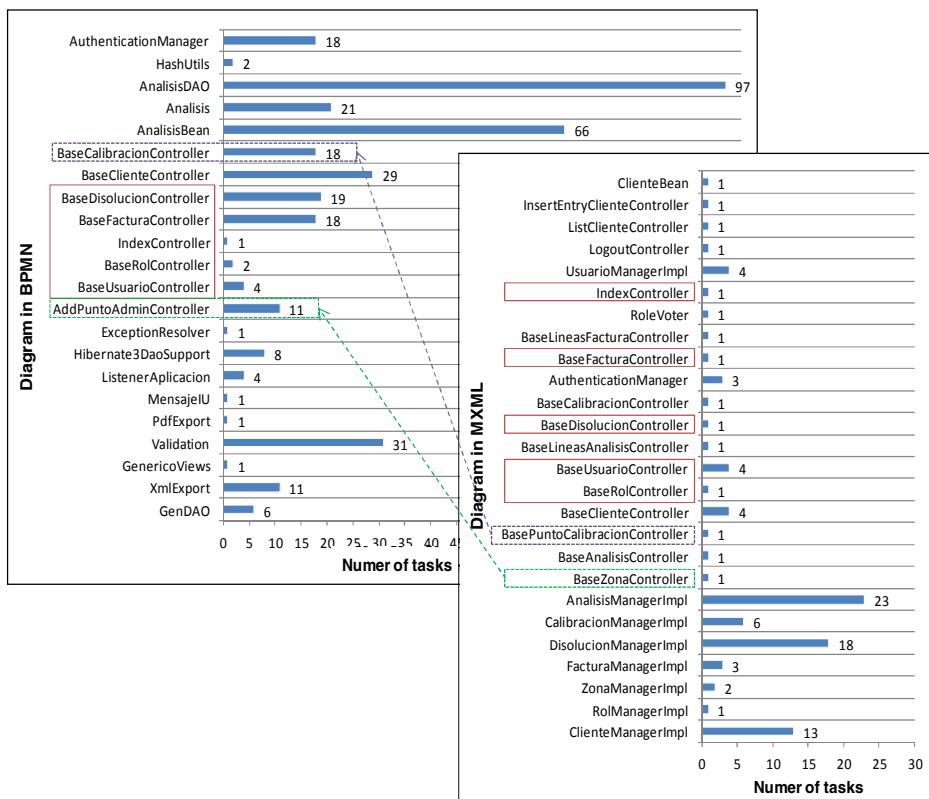


Fig. 2. Comparison BPMN and MXML: Diagrams framed in solid line are similar. Dotted MXML diagrams are tasks in dotted BPMN diagrams.

Thus, step 2 selects tasks that are similar in process model and event logs. The step uses the information obtained in the previous step to calculate the distance between two tasks of the two artifacts. If the MXML task is contained in a fine-grained diagram the distance between the BPMN task and MXML diagram is calculated.

The syntactic similarity is calculated using the *Levenshtein* distance [19] of the labels as Algorithms 1 to 3 show.

Algorithm 1. Obtaining Similar Tasks.

```

1  getSimilarTasks(Info InfoBPMN, Info InfoMXML)
2  List similarTasks;
3  for(tb:InfoBPPMN.getTasks()) do
4    for(tm:InfoMXML.getTasks()) do
5      if((getSimilarity(tb,tm))>=LIMIT) then
6        similarTasks.add(tb, tm);
7        if(tm.getDiagram().getType() == FINE_GRAINE) then
8          if ((getSimilarity (tb,tm.getDiagram()))>=LIMIT) then
9            similarTasks.add(tb, tm.getDiagram());
10   return similarTasks;
```

Algorithm 2. Obtaining the syntactic similarity between task names

```

1  getSimilarity (Task t1, Task t2)
2  double similarity;
3  double distance = LevenshteinDistance(t1.name, t2.name);
4  similarity = 1 - distance/max(t1.name.length, t2.name.length)
5  List adjacentT1 = getAdjacent(t1);
6  List adjacentT2 = getAdjacent(t2);
7  for(at1: adjacentT1)do
8    for(at2: adjacentT2)do
9      similarity += (getSimilarity(at1, at2)/10);
10   return similarity;
```

Algorithm 3. Obtaining the similarity between a task and a diagram

```

1  getSimilarity (Task t, Diagram d)
2  double distance= LevenshteinDistance(t.name, d.name);
3  return 1 - distance/max(t.name.length,d.name.length);
```

Table 2. Extract of Similar Tasks

BPMN Task	MXML Task	Similarity
BaseZoneController	setZoneManager	1
getUserManager	getUserManager	1
getRolManager	setRolManager	0.923076923
BaseRolController	setRolManager	1
BaseInvoiceController	setInvoiceManager	1
searchZoneNoHistoricas	searchZoneNoHistoricas	1
searchZone	searchZone	1
filterUser	filterUser	1
searchTypeAnalysis	searchTypeAnalysis	1
searchPointCalibration	searchPointsCalibration	0.956521739
searchLinesDissolution	searchLinesDissolution	1
vote	vote	1
authenticate	authenticate	1

After applying step 2 and following up with the running example, 45 similar tasks were detected, as shown in Table 2. MXML tasks in bold symbolize that these tasks are contained in a fine-grained process which are related to BPMN tasks. In this case, the used limit for the similarity between tasks is 0.9.

3.3 Step 3: Obtain Missing Sequence Flows to Be Added

This step uses the Heuristics Net obtained using PROM tool [18] and the set of similar tasks to determine which edges are candidates to be inserted. The source and target of an edge must be in the same diagram in the BPMN. Algorithm 4 shows the procedure used in this step. For each edge, the source and target task are searched from the set of similar tasks (line 5-6). If there are BPMN tasks similar to both tasks (source and target), then the occurrence of BPMN tasks included in the same BPMN diagram are checked (line 11). If the MXML target task has no similar BPMN task (line 14), an intermediate task is then searched (line 15-16). In this case, the obtained edge is induced by transitivity. Similarly, if the MXML source task has no similar BPMN task (line 25), an intermediate task is also searched (line 26-27).

To follow up with the running example, in this step 145 edges are studied, obtaining from Heuristics Net. After applying the third step 14 direct edges and 11 edges are transitively obtained as is shown in Table 3. However, edges with reflexive flows (same source and target) are not inserted in the model since they do not provide additional semantics.

Algorithm 4. Obtaining Edges to insert.

```

1 getSimilarEdges(HeuristicsNet h, similarTasks)
2   List similarDirectEdges;
3   List similarInducedEdges;
4   for(edge: h.getEdges()) do
5     List similarBPMNsources = getBPMNSimilar(edge.source, similarTasks);
6     List similarBMNTtarget = getBPMNSimilar(edge.target, similarTasks);
7     if(!similarBPMNsources.isEmpty()) then
8       if(!similarBMNTtarget.isEmpty()) then
9         for (Task source: similarBPMNsources) do
10           for(Task target: similarBMNTtarget) do
11             if(source.getDiagram()==target.getDiagram() &&
12                source!=target) then
13               similarDirectEdges.add(new Edge(source, target));
14             else then
15               for(intermediateEdge: h.getEdges()) do
16                 if(intermediateEdge.source == edge.target) then
17                   List similarBMNTtarget=
18                     getBPMNSimilar(intermediateEdge.target,similarTasks);
19                   if(!similarBMNTtarget.isEmpty())then
20                     for (Task source: similarBPMNsources) do
21                       for(Task target: similarBMNTtarget) do
22                         if(source.getDiagram()==target.getDiagram() &&
23                            source!=target) then
24                           similarInducedEdges.add(new Edge(source, target));
25             else then

```

```

26   for(intermediateEdge: h.getEdges()) do
27     if(intermediateEdge.target == edge.source) then
28       List similarBPMNtarget=getBPMNSimilar(intermediateEdge.target,
29         similarTasks);
30       if(!similarBPMNtarget.isEmpty())then
31         for (Task source: similarBPMNsources) do
32           for(Task target: similarBPMNtarget) do
33             if(source.getDiagram()==target.getDiagram() &&
34               source!=target) then
35               similarInducedEdges.add(new Edge(source,target));
36   return similarDirectEdges, similarInducedEdges;

```

3.4 Step 4: Insert Missing Sequence Flows

In the last step, the edges obtained in the previous step (see Table 3) are added to the process model. For each edge, its source task and its target task are located in the diagram and the sequence flow between both of these does not exist, the sequence flow is added. In the running example, 25 sequence flows (SF) are inserted in the process model since none of these previously existed.

Table 3. Sequence Flows to insert

	BPMN Source Task	BPMN Target Task	BPMN Diagram
Direct Sequence Flows	getAnalysisManager	BaseAnalysisController	AnalysisBean
	BaseAnalysisController	getAnalysisManager	AnalysisBean
	getCalibrationManager	BasePointCalibrationController	BaseCalibrationController
	BasePointCalibrationController	getCalibrationManager	BaseCalibrationController
	IoHandle	IoPrepareView	AnalysisBean
	nitBinder	IoHandle	AnalysisBean
	getDissolutionManager	BaseDissolutionController	BaseDissolutionController
	BaseDissolutionController	getDissolutionManager	BaseDissolutionController
	getInvoiceManager	BaseInvoiceController	BaseInvoiceController
	BaseInvoiceController	getInvoiceManager	BaseInvoiceController
	getRolManager	BaseRolController	BaseRolController
	BaseRolController	getRolManager	BaseRolController
Transitive Sequence Flows	getZoneManager	BaseZoneController	AddPointAdminController
	BaseZoneController	getZoneManager	AddPointAdminController
	PaginateAnalysisFiltered	searchTypeAnalysis	AnalysisDAO
	insertAnalysis	searchTypeAnalysis	AnalysisDAO
	searchAllClient	searchZone	AnalysisDAO
	searchPointSample	PaginateDissolutionsFiltered	AnalysisDAO
	searchTypeAnalysis	searchAllClient	AnalysisDAO
	searchZone	searchPointSample	AnalysisDAO

4 Case Study

This section provides a case study concerning Villasante-Lab, in particular the system presented in the running example. The case study has been conducted following the formal protocol developed by Runeson *et al.* [20] for conducting case studies in the software engineering field. The following sections show the stages of this protocol: the design, selection procedure, execution procedure and data collection, analysis and interpretation, and finally, the threats to validity.

4.1 Case Study Design

The *object* of this study is the proposed repair technique, while the *purpose* is the evaluation of its effectiveness in a real-life context in terms of accurateness and completeness. The following research questions (RQ) are established in order to carry out the case study:

- RQ1: Are repaired business models more accurate than preliminary models obtained by reverse engineering from source code?
- RQ2: Are repaired business models more complete than preliminary models obtained by reverse engineering from source code?

The case study follows the *embedded* case study design according to the classification proposed by Yin [21], since the case study consists of multiple units of analysis. The independent variables used in this study are business processes models. As dependent variables, conformance checking techniques are used in order to measure the fit degree between event logs and the target business process model after applying the technique. Conformance checking compares the observed and modeled behavior (i.e., event log). Hence, to answer the question RQ1, the dependent variable is the *fitness* value which is often seen as the most important quality dimension for comparing model and log [17, 22]. The fitness values vary between 0 and 1. A model has a perfect fitness (i.e., 1) if each trace in the event log can be replayed by the process model from beginning to end. To address question RQ2, as independent variable the *density* of the business process model is used, i.e., the ratio of the total number of edges in a process model to the theoretically maximum number of edges. The density, after inserting sequence flows, can only increase therefore this evaluation shows what to extent in a realistic case. RQ1 and RQ2 are therefore evaluated by means of quantitative research together with a qualitative evaluation, which focuses on the effectiveness of the proposed repair technique.

4.2 Case Selection Procedure

In order to select the case under study the following set of selection criteria are formulated: (1) the system should be a real-life information system currently in production; (2) the size of the system should be greater to 20 KLOC (thousands of lines of source code) to make it more likely that the system under study supports more than a single business process; (3) the system should be written in Java language to be able to use the supporting tools (MARBLE and Event Traces Injector).

After analyzing a dozen of information systems of partner companies according to criteria, the selected case was Villasante-Lab, a web application of 26 KLOC devoted to support operations of a chemical laboratory of the water and waste industry.

4.3 Execution Procedure and Data Collection

The procedure to carry out the case study consists of the following steps according to the proposed technique. Particular details of the execution are shown in the running example developed throughout Section 3.

1. Business process models are mined from the source code using MARBLE.
2. Event logs are obtained using the Event Traces Injector.
3. The repair technique is applied using the artifacts generated according to the described steps. In order to facilitate its execution, the technique has been implemented as a plug-in in the PROM tool.
4. The fitness in both business process models – the original from information and repaired using the proposed technique – is measured using the replayer proposed by *Adriansyah et al.* [22]. This technique is developed as a plug-in in the PROM tool. The fitness value is collected to carry out the conformance checking.
5. After the whole execution, the collected information is statistically analyzed to answer the research questions.

Table 4. Case Study's statistics

BP model	#tasks	Initial		Final		#inserted SF	Density gain
		Density	#SF	Density	#SF		
GenDAO	6	0.1333	2	0.1333	2	0	0
XmlExport	11	0.0355	60	0.0355	60	0	0
GenericViews	1	0	0	0	0	0	0
Validation	31	0.0448	35	0.0448	35	0	0
PdfExport	1	0	0	0	0	0	0
MessageIU	0	0	0	0	0	0	0
ListenerApplication	4	0.2222	2	0.2222	2	0	0
Hibernate3DaoSupport	8	0.1429	4	0.1429	4	0	0
ExceptionResolve	1	2.0000	0	2.0000	0	0	0
AddPointAdminController	11	0.0175	3	0.0292	5	2	0.0117
BaseUserController	4	0.0667	1	0.0667	1	0	0
BaseRoleController	2	0.1667	1	0.5000	3	2	0.3333
IndexController	0	0	0	0	0	0	0
BaseInvoiceController	18	0.1087	28	0.1159	30	2	0.0072
BaseDissolutionController	19	0.1082	25	0.1169	27	2	0.0087
BaseClientController	29	0.1261	80	0.1261	80	0	0
BaseCalibrationController	18	0.0627	27	0.0650	29	2	0.0023
AnalysisBean	66	0.0382	255	0.0386	259	4	0.0004
Analysis	21	0	0	0	0	0	0
AnalysisDAO	97	0.0265	124	0.0279	135	11	0.0014
HashUtils	2	0.6667	0	0.6667	0	0	0
AuthenticationManager	18	0.0342	4	0.0342	4	0	0
TOTAL	368	0.0096	651	0.0100	676	25	0.0004

4.4 Analysis and Interpretation

After the full execution of the case study, the values of the fitness were collected for the business process model. Although missing sequence flows were only detected in seven business process diagrams, as Table 3 shows, the fitness was calculated for the whole process model. The results demonstrated that the fitness of the repaired BP model (0.6064) is greater than the original fitness (0.3804), i.e., the repaired model fits 59.41% better to the observed behavior. However, the fitness is not yet close to 1 since, as was shown in Section 3.2: only 26% of business tasks of the whole information system are captured in the event logs.

Table 4 summarizes the statistics of the case study. Once the BPMN, the MXML and the Heuristic Net were available, the total time spent on carrying out the repair was 973 milliseconds. In all the cases the density gain (final density - initial density) was positive, even reaching a 33.33% gain.

Hence, the research question RQ1 may be positively answered owing to the fitness has increased, i.e., the repaired business models are more accurate than the preliminary model obtained by reverse engineering from source code. Similarly, the research question RQ2 may be positively answered since the final model is more connected and therefore more complete.

4.5 Threats to Validity

This section presents the threats to the validity of this case study and possible actions to address them. The threats are divided in three types of validity: internal, construct and external validity.

Regarding the internal validity, the study considers a process model and event logs obtained from an information system. However, the study may be replicated by using more information systems, to consider a larger sample of process models. Besides, the support tools (MARBLE and ETI) could be a factor that affects the case study results since the technique depends on the settings of retrieved process model and event logs.

With regard to construct validity, the study considers measures to evaluate the research question. Nevertheless, there are other measures in literature that may be used instead. Hence, additional measures should be evaluated in the future, such as shown in [3]. Another threat to construct validity is the similarity algorithm used in step 2 to obtain similar tasks (Algorithm 1). In order to address this threat, other possible similarity algorithms may be considered as e.g., including the semantic similarity.

Concerning the external validity, this study considers the whole population to be business process models retrieved by reverse engineering from legacy information systems as well as event logs obtained from the same information system. However, the obtained results obtained cannot be strictly generalized to all types of information. This threat may be mitigated by replicating the study using systems implemented in different platforms.

5 Conclusions and Future Work

Reverse engineering has become a feasible solution to mine business processes models from existing information systems. Unfortunately, these retrieved business processes models entail some challenges that are necessary to address if synch models form the basis for properly managing these business processes.

Incompleteness is one such important challenge to deal with in a retrieved business processes model, since data are distributed across several sources. Missing sequence flows between elements decreases the understandability of the model since it may not reflect the real behavior of an organization. In order to address this challenge, this paper present a technique for repairing business processes models obtained from information systems using event logs. The technique builds on two assumptions: (1) business process models, which represent the static viewpoint of the organization, are mined by the archeology tool MARBLE, which is an adaptive framework to recover business process models underlying legacy information system; and (2) event logs, which represent the dynamic viewpoint of an organization, are obtained by means of the technique proposed in [8], since event logs cannot automatically be generated from non-process-aware systems. Despite these assumptions, the main ideas of this approach can be easily adapted to other reverse engineering techniques and platforms. In fact, to ensure its feasibility this technique has been validated by means of an industrial case study. The results of this case study show that the fitness of the process model increases after applying the technique, i.e., repairing business process model leads to a more faithful representation of the observed behavior.

Future work will aim at incorporating a mechanism to calculate the semantic distance between two tasks. Besides, with both mechanisms (syntactic and semantic similarity) can be performed a grouping of similar tasks in order to decrease the number of fine-grained tasks, i.e., those tasks that do not perform a real business activity. Finally, a mechanism is called for to detect tasks' labels which are poor in descriptive quality, i.e., those task labels that have several occurrences in the model and do not clearly represent their purpose.

Acknowledgments. This work was supported by the FPU Spanish Program and the R&D projects GEODAS (TIN2012-39493-C03-01), PEGASO /MAGO (TIN2009-13718-C02-01) and ARMONIAS (PII2I09-0223-7948). This work has been additionally supported by Eindhoven University of Technology.

References

1. Weske, M.: *Business Process Management: Concepts, Languages, Architectures*. Springer, Heidelberg (2007)
2. Pérez-Castillo, R., García-Rodríguez de Guzmán, I., Piattini, M.: *Business Process Archeology using MARBLE*. Information and Software Technology (2011)
3. Fernández-Ropero, M., Pérez-Castillo, R., Caballero, I., Piattini, M.: *Quality-Driven Business Process Refactoring*. In: International Conference on Business Information Systems, ICBIS 2012, Paris, France, pp. 960–966 (2012)

4. Fahland, D., van der Aalst, W.M.P.: Repairing Process Models to Reflect Reality (2012)
5. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
6. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., de Medeiros, A.K.A., Song, M., Verbeek, H.: Business process mining: An industrial application. *Information Systems* 32, 713–732 (2007)
7. Dumas, M., van der Aalst, W.M.P., Ter Hofstede, A.: Process-aware information systems. Wiley Online Library (2005)
8. Pérez-Castillo, R., Weber, B., García Rodríguez de Guzmán, I., Piattini, M.: Generating Event Logs from Non-Process-Aware Systems Enabling Business Process Mining. *Enterprise Information System Journal* 5, 301–335 (2011)
9. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering* 16, 1128–1142 (2004)
10. De Medeiros, A.K.A., Weijters, A., van der Aalst, W.M.P.: Using genetic algorithms to mine process models: representation, operators and results. Beta, Research School for Operations Management and Logistics (2005)
11. Weijters, A., van der Aalst, W.M.P., de Medeiros, A.K.A.: Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Tech. Rep. WP 166 (2006)
12. Zou, Y., Hung, M.: An Approach for Extracting Workflows from E-Commerce Applications. In: Proceedings of the Fourteenth International Conference on Program Comprehension, pp. 127–136. IEEE Computer Society (2006)
13. Ghose, A., Koliadis, G., Chueng, A.: Process discovery from model and text artefacts. In: 2007 IEEE Congress on Services, pp. 167–174. IEEE (2007)
14. Adriansyah, A., van Dongen, B., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis, pp. 55–64. IEEE (2011)
15. ISO/IEC: ISO/IEC 19506:2012. Information technology – Object Management Group Architecture-Driven Modernization (ADM) – Knowledge Discovery Meta-Model (KDM), p. 331. ISO/IEC (2012)
16. <http://www.omg.org/spec/BPMN/2.0/PDF/>
17. van der Aalst, W.M.P.: Process Mining: Overview and Opportunities. *ACM Transactions on Management Information Systems (TMIS)* 3, 7 (2012)
18. Promtools.org: ProM Tool (2010)
19. Lvenshtein, V.: BINARY coors CAPABLE or ‘CORRECTING DELETIONS, INSERTIONS, AND REVERSALS. In: Soviet Physics-Doklady (1966)
20. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Eng.* 14, 131–164 (2009)
21. Yin, R.K.: Case study research. Design and methods. Sage, London (2003)
22. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replay history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2012)

Bridging Abstraction Layers in Process Mining: Event to Activity Mapping

Thomas Baier¹ and Jan Mendling²

¹ Business Process Technology Group

Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3, D-14482 Potsdam, Germany
thomas.baier@hpi.uni-potsdam.de

² Wirtschaftsuniversität Wien, Augasse 2-6, 1090 Vienna, Austria
jan.mendling@wu.ac.at

Abstract. While the maturity of process mining algorithms emerges and more process mining tools enter the market, process mining projects still face the problem of different levels of abstraction when comparing events recorded by supporting IT systems with defined business activities. Current approaches for event log abstraction most often try to abstract from the events in an automated way which does not capture the required domain knowledge to fit business activities. This can lead to misinterpretation of discovered process models and wrong conformance results. We developed an approach which aims to abstract an event log to the same abstraction level which is needed by the business. Therefore, we capture domain knowledge about event to activity mappings in a formalized way and propose an algorithm to correctly cluster events to activity instances. We evaluated our approach in a case study with a German IT outsourcing company.

Keywords: Process Mining, Abstraction, Event Mapping.

1 Introduction

Process mining is an emerging research field which gets more and more relevant for application in praxis due to its increasing maturity. Using the event data logged by IT systems, process mining algorithms discover and enhance process models or check whether the execution of a process conforms to specification [1]. Looking at conformance checking and enhancement, it is obvious that the events stemming from the IT system have to be mapped to the activities defined in the process models. However, the events are typically more fine granular than the activities defined by business users. This implies that different levels of abstraction need to be bridged in order to conduct a conformance analysis. Furthermore, such a mapping is not only necessary for conformance checking and process model enhancement, but also for discovery. The benefit of a discovered process model can only be fully exploited if the presented results are on an abstraction level which is easily understandable for the business user. Nevertheless, most of current process mining techniques assume that there is a 1:1 mapping

between events and activities. There are some abstraction approaches which try to close this gap by preprocessing the event log and automatically finding clusters of events which can be bundled into activities. Yet, these techniques have limited capabilities when dealing with concurrency and n:m relations between events and activities and give no or only limited possibilities to correctly refine such mappings based on domain knowledge.

The contribution of this paper is a systematic definition of this mapping problem as well as a mapping approach which uses encoded domain knowledge. We provide the different possibilities to map events to activities and define strategies for merging event instances into activity instances. In contrast to existing approaches, the method introduced in this paper is designed to deal with concurrency and to handle n:m relations between events and activities. As there is no automatism involved in the definition of mapping between events and activities, there are no specific requirements to the event log in terms of size or variance. This event log abstraction approach can be used as a preprocessing for every process mining technique. Furthermore, we present results from using different strategies in a service management case study. The results emphasize the sensitivity of conformance checking to the defined mapping problem.

The paper is structured as follows. Section 2 describes the problem of different abstraction levels in event logs and process models. Coming from the problem definition, section 3 introduces the preliminaries for our approach and finally, the strategies to overcome the gap between abstraction levels of event log and process model. These are applied to event logs in section 4. In section 5, we show the results from a case study of a German outsourcing provider where we used different mapping strategies in order to compare their fitness and to outline the implications on conformance testing. Related work is discussed in section 6 and section 7 concludes the work.

2 Background

In this section we illustrate the problem at hand. First, we will look at the problem *from the perspective of the process model*. In order to illustrate the different abstraction layers on design and event log level, Fig. 1 shows a simple process model with a sequence of activities A, B and C. For each activity we find a set of related events in the event log. Hence, the activities and events are in a 1:n relation which is due to the fact, that the event log is on a lower level of abstraction compared to the designed model. Thus, there has to be a bridging model in between which is on the same level as the event log and shows the relations between the events and their relations to the high level activities defined in the process model. Yet, this model is typically not available and too complicated to be created. One possible model of these sub activities is shown in Fig. 1. Here, we assume that all events are mandatory and occur in a strict sequence. Nevertheless, every other order of these events is possible and could also include loops, parallelism or exclusions. Furthermore, certain events might not be mandatory. Note that Fig. 1 also shows that events with the same name might not always belong to the same activity because they represent some shared

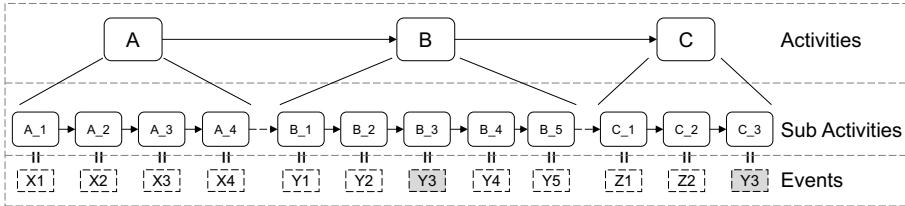


Fig. 1. Abstraction levels with shared functionality: From design to event log

functionality which is used in different activities. An example of such a shared activity is the writing of protocols during the execution of an activity. This is often done for several activities in a process. The bridging model would specify the process in that detail that each sub activity could be identified with an event. In the absence of this bridging model, the relationship between events belonging to the same activity remains unspecified.

Second, we can look at the problem *from the perspective of the event logs*. Here, the direction of mapping points from event instances towards identifying more abstract activity instances. Figure 2 shows the different levels of abstraction which have to be taken into account and illustrates two different mapping strategies which lead to different discovered process executions. On the bottom line we have the unaltered events which have been extracted from the supporting IT system. On the top of Fig. 2 the designed process is shown. In the first step each event reported by the IT system is mapped to an activity from the designed process model. Applying the mapping to the event log results in a new event log of the same size where all events have been renamed with the names of the corresponding activities they have been mapped to.

The fact that multiple event classes may belong to one activity again reflects that event log and process model do not have the same level of abstraction. Thus, the mapped event log contains duplicate entries where different events have been mapped to the same activity name. Looking at the example in Fig. 2, we have four instances of activity A in the mapped event log. With the knowledge of the sequential process model we could assume, that each occurrence of A belongs to one activity instance as long as no other activity occurs in between. Thus, we might group all adjacent occurrences of A as shown in Fig 2 (a). This again leads to several instances of activity A. As we do not know whether the process in reality follows the sequential model, we could also take the approach to merge all event occurrences of one activity into one activity instance as shown in Fig. 2 (b). Using this strategy raises the problem that we eliminate all loops and rework from the event log. Hence, both strategies might not be realistic in general. The question that we aim to investigate is what reasonable strategies for abstraction are, which aggregate an event log to the same level as a given set of business activities. Therefore, we investigate the mapping of events to business activities and the clustering of events to activity instances.

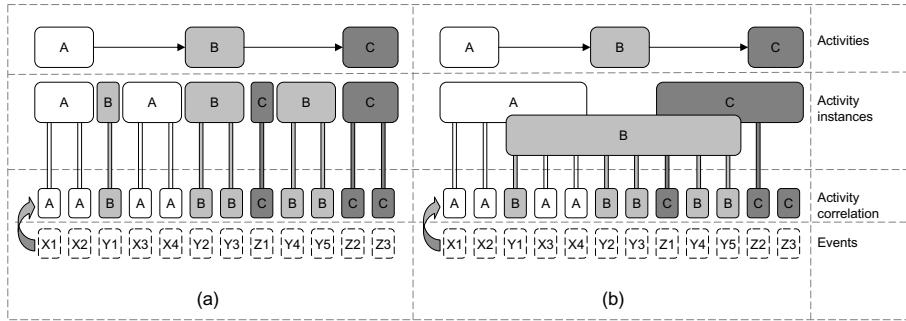


Fig. 2. Different abstraction results on the instance level

3 Abstraction Approach

3.1 Preliminaries

In this section we introduce the preliminaries on which we ground our work. This includes the definition of a process model, a process execution and an event log. Definition 1 formalizes a process model as a tuple of activities, control nodes and flow relation. Based on this, we introduce the notion of a process execution in Definition 2. Note that this definition also includes process executions which do not conform to the process model.

Definition 1. (*Process model*) A process model is a triplet $P = (A, F, C)$, with

- A as a non-empty set of activities,
- C as a set of control nodes which is disjoint to set A
- $F \subseteq (A \cup C) \times (A \cup C)$ as the flow relation, which connects activities with each other and with control nodes to build a directed graph.

Definition 2. (*Process execution, process instance, activity instance*) Let P be a process model and A the set of defined activities within this model. A process execution is any sequence of activity instances $\hat{a} \in \hat{A}^*$.

An IT system which supports process executions typically records events in an event log. Definition 3 presents the basic concepts of an event log.

Definition 3. (*Event class, event instance, event attributes, trace, event log*) Let E be the non-empty set of event classes and \hat{E} the set of event instances. Each event instance has different attributes ($attr \in AT$) assigned to it. For each attribute $\#_{attr}(\hat{e})$ refers to the value of the attribute $attr$ for the event \hat{e} . A trace is a list of the form \hat{E}^* . An event log L is a multi-set of traces over \hat{E} , i.e., $L \in \mathbb{B}(\hat{E}^*)$.

We assume that an event has the following standard attributes (cf. [1]):

- $\#\text{name}(\hat{e})$ refers to the event name and thus, to the event class.
- $\#\text{time}(\hat{e})$ refers to the time when the event occurred.
- $\#\text{trans}(\hat{e})$ refers to the transaction type of the event, i.e. whether the event has been started or completed.

Furthermore, each trace in an event log is related to a process execution. Note that the formerly described attributes are only the mandatory attributes we assume for every event. Other attributes may contain role information or values of changed database field which might also be used for defining the mapping.

3.2 Event to Activity Mapping

In order to lift an event log to the abstraction level of a corresponding set of activities, we define a mapping relation M which assigns every event instance $\hat{e} \in \hat{E}$ to an activity instance $\hat{a} \in \hat{A}$. Hence, all events in an event log have to be either mapped onto their corresponding activity or removed from the log.

Definition 4. (*Mapping function*) Let $P = (A, F, C)$ be a process model which models the process that has been recorded in event log $L \in \mathbb{B}(\hat{E}^*)$. Then $M : \hat{E} \rightarrow \hat{A}$ is the mapping which assigns every event instance from the event log to an activity instance of a valid or non-valid process execution of P .

Although the mapping function M is defined on instance level, we have to take both dimensions into consideration: the class level and the instance level. Looking at these two dimensions, we furthermore have to take into account the cardinality of the relationship between A and E as well as between \hat{A} and \hat{E} . Following these aspects, we can distinguish the different types of mappings as shown in Table 1

Table 1. Types of mappings on class level

Relation	Class level
1:1	Exactly one event class represents one activity.
1:n	One activity is represented by different event classes due to lower level of abstraction in the event log.
n:1	Multiple activities are reflected by the same event class because a certain subactivity is performed in more than one activity.
1:0	An activity is not recorded in the event log.
0:1	Event class is out of scope and cannot be mapped to an activity.

Table 2. Types of mappings on instance level

Relation	Instance level
1:1	One activity instance consists of exactly one event instance.
1:n	An activity instance can be represented by multiple event instances.
n:1	One event instance reflects instances of multiple activities.
1:0	An activity has not taken place.
0:1	Event instance cannot be mapped to an activity instance.

and Table 2. An event to activity mapping is always a combination of both a mapping on class and a mapping on instance level.

The handling of activities which are not recorded in the log is out of scope of this paper. Thus, we don't elaborate on 1:0 relations. Considering 0:1 relations, there is simply no mapping and the events have to be removed from the event log. Regarding the mapping of 1:1 relations on class level, we can simply rename event classes to match the names of their corresponding activities and rename event instances respectively. In order to illustrate this with the mapping of an example trace, we assume a 1:1 relation on instance level. Thus, if we assume a 1:1 relation on both levels between event class x and activity a the trace $\hat{x}\hat{x}\hat{x}\hat{x}$ would simply map to the trace $\hat{a}\hat{a}\hat{a}\hat{a}$.

The mapping of 1:n relations on class level is also straight forward. Assuming the trace $\hat{x}_1\hat{x}_2\hat{x}_2\hat{x}_3$ and the class level mapping $x_i \rightarrow a$ we get the mapped trace $\hat{a}\hat{a}\hat{a}\hat{a}$ if we assume a 1:1 mapping on instance level. Looking again at the trace $\hat{x}_1\hat{x}_2\hat{x}_2\hat{x}_3$, it might also be the case that the second execution of x_2 refers to a different activity than the first execution. This means that on class level we have multiple activities reflected by the same event class, i.e. a n:1 relation on class level. Here, the context of the event instances has to be considered to make a decision. The context can either be defined over the event attributes or the surrounding event instances. Hence, depending on the role by which an event has been recorded this might refer to a different activity in the process model. For example a protocol entry by an expert might refer to an activity called “deep analysis” while a protocol entry by a novice might refer to an activity called “pre-analysis”. Similarly, an event might be interpreted differently if it occurs for the first time or if it has been preceded by earlier executions. Taking again the protocol example, it could also be the case that a deeper analysis has taken place when the event has been recorded for the second time. In order to use such domain knowledge, we have to encode it in a formal way. Definitions 5 and 6 introduce the formalization of meta data conditions and event context conditions.

Definition 5. (*Meta data condition*) Let O be the set of comparison operators, e.g. $O = \{\text{equals}, \text{contains}, \text{startswith}\}$, and let V be the set of values that an event attribute $\text{attr} \in AT$ should be tested against. Then, a meta data condition is a tupel $mdc \subseteq AT \times V \times O$.

Definition 6. (*Event context condition*) An event context condition is defined by a tupel $ecc = (f, r)$, where f is a condition defined as linear temporal logic formula¹ and $r \in (\text{before}, \text{after})$ refers to the part of an event context \hat{EC} in which this formula should be tested. The event context \hat{EC} is a tupel $(t_{\text{before}}, t_{\text{after}})$ where t_{before} and t_{after} are substraces of a trace t such that $t = t_{\text{before}} \parallel \hat{e} \parallel t_{\text{after}}$.

To map event instances to activities, we define an event class to activity mapping EAM based on event classes and conditions that have to be fulfilled for a corresponding event instance in order to be mapped to a specific activity. The conditions can be meta data conditions as well as event context conditions.

¹ See [2] for detailed information on linear temporal logic (LTL).

Definition 7. (*Event class to activity mapping*) An event to activity mapping is a function $EAM : E \times MDC \times ECC \rightarrow A$, which relates an event class to an activity class based on a set of meta data conditions and a set of event context conditions.

Definition 7 gives the mapping between activities from a process model and event instances found in an event log. For the protocol example the role based mapping could be defined as

$('protocol', \{('role', 'specialist', 'equals')\}, \{\}) \rightarrow' deepanalysis'$

and the occurrence based mapping could be defined as

$('protocol', \{\}, \{(<> ('protocol'), 'before')\}) \rightarrow' deepanalysis'.$

Coming from such a mapping, we define a function $\hat{E}AM$ which maps event instances surrounded by an event context to an activity.

Definition 8. (*Event instance to activity mapping*) $\hat{E}AM : \hat{E} \times \hat{EC} \times EAM \rightarrow A$ is the function which maps an event instance \hat{e} to an activity class A based on the event context EC and the given event class to activity mapping.

Having a mapping function $\hat{E}AM$, the next step is to define how to map event instances belonging to the same activity class to activity instances. As there might be multiple activity instances for one activity in a process execution, i.e. in a loop, the question is which criteria are used to map an event instance \hat{x}_i to an activity instance \hat{a}_j . Hence, we need to define where to draw the border between events belonging to two or more instances of the same activity. Definition 9 therefore introduces the notion of instance border conditions.

Definition 9. (*Instance border condition*) An instance border condition defines under which conditions two event instances which are mapped to the same activity a cannot belong to the same activity instance \hat{a}_i and thus, represent two different activity instances \hat{a}_i, \hat{a}_j . It is defined as a boolean function $BC : \hat{E} \times \hat{E} \rightarrow \{\text{true}, \text{false}\}$.

Instance border definitions are essentially statements about the behavioral structure of the assumed sub activity model, i.e. whether there are loops over sub activities or not. Note, that we are referring to the assumed sub activity model, not to the actual sub activity process captured in the event log. While the assumed sub activity model might not contain loops, this does not imply that there are no loops in the execution on sub activity level. If there are loops on execution level recorded in an event log, they will be lifted to activity level if we assume there should not be any loops on sub activity level.

Stating that there are no loops in the assumed sub activity model, an activity instance border is marked by the repetition of source events from the same event class, i.e. the repetition of a source event class signals that a new activity instance has started. Thus, for example two protocol events could indicate rework and therefore two instances of the corresponding activity.

Using recurring source event classes as instance border definition works only if there are no loops in the assumed sub activity model. If there are loops in the

assumed sub activity model and in the process model, multiple event instances from the same event class might belong to one activity instance. A typical example for this is a loop over the order items in an order list where different activities like “Choose supplier” and “Sent order” have to be performed for each order item and are modeled in the process model on activity level. The sub activity “Choose supplier” might again contain different activities which have to be executed for each supplier, like e.g. “Check prices”. Thus, we have a loop on activity level and a loop on sub activity level. In order to find the correct instance borders, we need to extend the instance border definition to also use different business objects, e.g. the order line, as instance border markings. Thus, instance borders can be defined over any meta data attached to an event.

If such meta data is not available or if it is not possible to make such statements about the assumed sub activity model, we need to use heuristics in order to be able to identify different activity instances. One possible heuristic for instance border definitions is to make assumptions about the maximal number of events which belong to one activity instance. This is simple if we can exclude the occurrence of loops on sub activity level, but might be difficult otherwise.

Another heuristic which is more independent from the sub activity model, can be defined as a threshold for the maximal distance between two events which belong to one activity instance. This distance can be specified using the time perspective, i.e. defining how long the time frame between two events of the same activity instance can be. For example one might limit the time distance between two events of the same activity instance, e.g. two edit events for a protocol belong to different activity instances if there are more than a 24 hours between them. Another way to specify the distance is to us a maximal number of events which are allowed to occur between two events of the same activity instance.

Definition 10 defines the function \hat{EAM} which uses the previously defined event instance to activity correlation provided by \hat{EAM} and maps these pairs of the form (\hat{e}, a) to activity instances using defined instance border conditions.

Definition 10. (*Event instance to activity instance mapping*) *The function \hat{EAM} provides a set of event instances mapped to activity classes which is a subset of $\hat{E} \times A$. Then $\hat{EAM} : \hat{E} \times A \times BC \rightarrow \hat{A}$ is the function which assigns a previously mapped event instance to its corresponding activity instance \hat{a} . For each $a \in A$ there is a set of instance border conditions $bc \subset BC$. An event instance \hat{e} which is mapped to an activity instance \hat{a} is referred to as source event of \hat{a} .*

Definition 10 covers 1:1 and 1:n relations on the instance level. A 1:1 mapping on instance level only occurs for events and activities which are on the same abstraction level. Looking at different abstraction levels it is most likely that an activity instance on a higher level subsumes multiple event instances representing different sub activities. Thus, in most cases we face a 1:n mapping on instance level and event instance will be clustered to activity instances. Nevertheless, it can be the case that one event instance reflects multiple activities. Coming again back to our protocol example, it might happen that the protocol does not only contain the analysis results but also the planning of the next steps. This might be recognized by scanning the content of the protocol which is attached to the

event instance as meta data. To solve this n:1 relation problem, we can simply duplicate the event instance to create a 1:1 or 1:n mapping on instance level.

4 Event Log Transformation

In the previous section we defined necessary functions and their inputs in order to map event instances to activity instances. Based on these definitions, this section elaborates on the technical transformation of a complete event log.

First, we iterate over the traces in a log and rename the event instances with the names of the activities assigned by the event class to activity mappings (EAM) which are to be provided by domain specialist. This leads us to the activity mapping stage as depicted in Fig. 2. As a consequence of the 1:n relation, we get duplicates. The challenge is now to find the event instances which belong to the same activity instance and cluster them. Hence, we need to find the borders of activity instances, marking the end of an instance or the beginning of new instance respectively as defined by the function \hat{EAM} .

The algorithm we propose for the clustering of events to activity instances is grounded on a tree-based incremental clustering algorithm known from classical data mining (cf. incremental clustering in [3]). For every activity class the clustering forms a tree with the event instances as leaves and the activity instances on the next higher level. The clustering starts with an empty tree for each activity class consisting of the root alone. Next, the event instances are incrementally inserted into the tree. Updating the tree is done by finding the right place to put the new leaf. This may also cause the restructuring of the part of tree that is affected by the new instance. The algorithm consists of 4 main steps:

- 1. Find the best host for a new event.** The best host for a new event is the host with the minimal distance. This distance can be expressed by a distance function d using the time perspective, the number of events in between, the calculated distance between other event attributes or a combination of these. In case we can exclude concurrent executions of the same activity class and do not include specific event attributes, the best host will always be the last inserted event of the same activity class as events are sorted chronologically in the event log.
- 2. Check if new event signals an activity instance border.** Having found the optimal host, we have to check for all events which belong to the same activity instance as the host event that these do not fulfill any border condition in combination with the event that has to be inserted into the cluster. Note that in this step we only check if an instance border exists. We do not yet know where this border is, if a border is found. The new event instance might already have started before, when an event is observed which signals an instance border. Step 3B will deal with finding the beginning of a new activity instance by optimizing the clustering tree.

3A. If no instance border is found, add event to cluster of host

3B. If instance border is found, find optimal cluster splitting. When a new activity instance is recognized, we need to evaluate to which of the instances the already assigned events belong. Therefore, a goodness measure g is introduced which enables us to make relative comparisons between different cluster possibilities. We define the goodness measure as the sum of the distances in the event log between the adjacent events of a cluster: $g_C = \sum_{i=1}^{C_{\text{size}}-1} d(C_i, C_{i+1})$. The goodness measure is not only defined for single clusters, but also for sets of clusters. For sets of clusters g is defined as the sum over the goodness measures of the encompassed clusters: $g_S = \sum_{j=1}^{S_{\text{size}}} g_{C_j}$. Hence, we can compare different clustering results with each other and choose the best. Smaller values for g signal better clusters as the events within the cluster are closer to each other.

The algorithm starts by adding the new event under a new activity instance. Next, event after event are moved from the host activity instance to the new activity instance. At each point in this iteration the goodness measures for the two activity instances and the root are calculated. It is checked that no instance border condition is fulfilled between any two events of an activity instance. Otherwise the cluster combination is discarded. If we exclude concurrent executions of the same activity class, it is sufficient to move event after event from the host cluster to the new cluster starting with the last inserted event and ending before the event is moved which fulfills the border condition. Otherwise, all combinations have to be tested which leads to exponential effort ($\mathcal{O}(2^n)$). In the end, the clustering with the lowest goodness value is chosen as the optimal cluster and gives us the border between the two activity instances.

We will explain the basic algorithm using an example. Fig. 3 shows an example event sequence. The mapping EAM for this example is defined so that every event class x_i maps to an activity x and every event class y_i maps to an activity y . Looking at the instance border conditions, we define that there are no loops on sub activity level. We further define that there are no loops and no parallel executions of instances of the same activity.

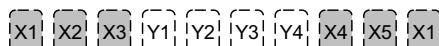


Fig. 3. Event sequence example

Figure 4 shows the clustering steps for activity x . Events are added to the cluster as long as they do not fulfill the defined border conditions. When adding the second instance of event x_1 , it is recognized that there is already an instance of x_1 in the current cluster. Thus, a border condition is fulfilled and a new cluster for a new instance has to be created (Fig. 4 (2)).

Looking at the clustering example in Fig. 4, the insertion of the second x_1 at first results in two clusters with $g_{C_1} = 8$ and $g_{C_2} = 0$. Next, the algorithm

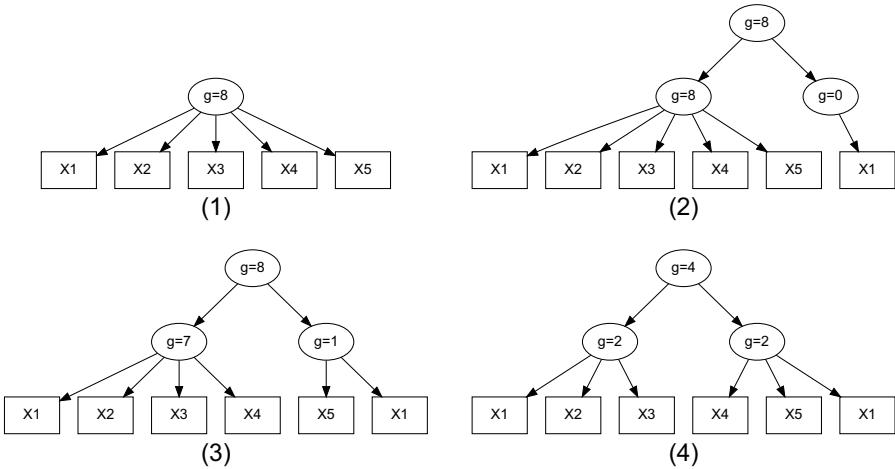


Fig. 4. Clustering events of activity class X to activity instances using tree structures

moves event after event from the first cluster to the second cluster until he finds the optimal solution. In the example, the optimal result with $g_C = 4$ contains the two clusters as shown in Fig. 3 (4).

5 Evaluation

In order to evaluate our approach with real life data, we did a case study in a large German IT outsourcing project. The process being under investigation is the service request and incident management process which is supported by the ticketing software IBM Tivoli Service Desk. The extracted event log contains 17,209 cases, 31 event classes and a total of 479,408 events for a selected month.

First, the event log mapping had to be created by the process manager of the process. The process model to which the event log should be mapped contains 39 unique activities and has been modeled as an event-driven process chain (EPC). The result mapping comprises 68 mapping definitions that follow the form of *EAM* as defined in Def. 7. 13 of these are simple string mappings while 14 include meta data conditions and 41 use event context conditions. Out of the 41 event context conditions, 6 conditions model the fact that an event belongs to a different activity when it occurs for the first time. An example for this is the event that records the change of the group to which an incident ticket is assigned. The first event reflects the initial setting of the group which created the ticket while all other changes of the group signal the routing of the ticket. Moreover, there is another distinction between the routing activities formulated in meta data conditions. Based on the group name to which an incident ticket is routed, the activity is either “Route to 1st level”, “Route to 2nd level”, or “Route security incident”. As there is no role information in the event log, most of the other event context conditions use these routing events to determine whether an

event has been created by a 1st level or by a 2nd level unit. Another example of event context conditions are the quality checks of certain ticket attributes after the solution has been found. Here, events that reflect for example the change of a connected configuration item are assigned to such a quality check activity if they occur after the solution of the incident.

In order to get good abstraction results, it is necessary to validate the chosen abstraction approach and to choose the right activity instance border definitions. For the purpose of validating our approach and to show the influence of different choices for instance border definitions, we implemented our approach as a plugin in the process mining workbench ProM.² The abstraction results are compared to a gold standard event log to judge their goodness. Therefore, a domain expert manually mapped the events of 152 cases to activities.

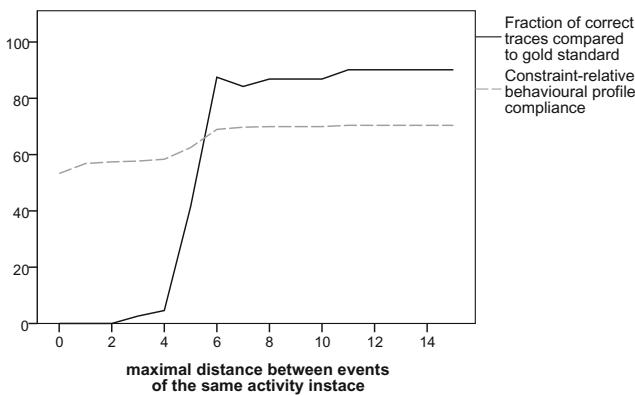


Fig. 5. Influence of different values for the heuristic instance border definition of maximal distance between two events belonging to the same activity cluster

First, we want to compare the abstraction results for different activity instance border definitions. We differentiate between merging (a) and not merging (b) event instances stemming from the same event class and hence, either allow for loops on sub activity level or not. As a third option, we consider that a single resource might do things iteratively and thus, events from the same event class and same resource belong to one activity instance while this is not the case if they are created by different resources (c). Option (b) returns with 52 % the least percentage of correct traces when compared to the gold standard. When allowing for loops on the sub activity level (a) this results in 90 % of all traces being correctly abstracted. The option which only allows loops executed by a single resource performs best and even yields 100 % correct cases. The reason for this is e.g. the iterative filling of fields in the web mask of the ticketing tool. If a service agent writes down the problem description and pushes the save button several times during this activity, there will be a new event for every saving. These events obviously belong to the same activity instance. In contrast, when

² See <http://processmining.org> for more information on ProM.

a different service agent picks up the ticket and needs to update the problem description, this is seen as rework and thus, as a new activity instance.

As explained before, it might not always be the case that we have the required meta data, such as the resource, available in the event log. In this case we have to use a heuristic. Figure 5 therefore shows the impact of different maximal distances between event instances belonging to the same activity instance. This reveals that there are up to 10 events between two events belonging to the same event instance. We can also see that this is a saturation point for the maximal merging distance and results are not changing with higher values anymore. Thus, in this scenario it is best, not to restrict the distance between two events at all. Doing so we retrieve a fraction of 90 % of all traces that are correctly abstracted. For different maximal cluster sizes we obtained a similar result.

To further assess the impact of different instance border definitions we investigated their influence on conformance analysis using the constraint-relative behavioral profile compliance metric as defined in [4]. Figure 5 shows the results for the heuristic activity instance border definition over event distances. We can see that there is a difference of 17 % points in the compliance metric between the lowest distance and the saturation point. Thus, there is a quite big influence of the correct activity instance clustering on conformance checking.

Summing up, we showed in this section that the presented approach performs well in a practical setting. Furthermore, we showed influences of instance border conditions on the abstracted result as well as on conformance analysis. It has to be noted that there is still a significant part of manual work to be done in order to encode domain knowledge into the required mapping definitions. Other approaches for event log abstraction require less manual work or are even fully automated, but in return are not able to fully capture the introduced concepts and thus, will lead to inferior abstraction results. We will discuss these approaches in the next section.

6 Related Work

Looking at the literature in the process mining area, there are different works on abstraction of event logs which are highly related to the approach presented in this paper. There are several works which try to automatically derive semantically related clusters of events in order to abstract these clusters to activities. Günther and van der Aalst [5] first developed an approach to cluster events to activities on the process and event instance level using a distance function based on time or position distance similar to the one presented in this paper. Using the distance function the authors build initial clusters which are refined by a configurable clustering function which might take into account e.g. modified data types and originators in order to cluster events belonging to the same activity. In [6], Günther et al. note that this approach has serious performance problems and introduce a new means of abstraction on event class level. Event classes are clustered globally by computing their relatedness in terms of co-occurrence over all process instances in an event log. This approach yields a higher performance

but lower quality as event classes are only allowed to belong to one activity. Li et al. [7] use a similar co-occurrence-based approach to find patterns of events which are supposed to be semantically related. In contrast to the work in [6], they allow for n:m relations between event classes and activities as they do consider the context of an event instance when mapping it to an activity. Yet, only events potentially belonging to the same activity are considered for the event context. Furthermore, there is no possibility yet to include meta data and the approach has a very sharp definition of activity instance borders, which does not allow more than two events between the events belonging to one activity instance. Thus, it faces problems when dealing with concurrency. Moreover, a significant amount of event log data is required in order to find reasonable patterns.

In contrast to the work in [5], [6] and [7], the approach presented in this paper is able to deal with concurrency and n:m relations from shared activities while delivering a good performance. What is more, it is independent from the size and variance of the available event log data. Even though the presented abstraction approach requires more manual work, it provides a transparent and simple means for the user to influence the abstraction result and has proven suitable in a practical case study.

While the formerly discussed approaches focus on clustering similar behavior, Greco et al. [8] developed an abstraction approach which clusters process instances with similar behavior and focus on abstracting behavior which is different between these clusters. While this is an interesting approach for exploratory analysis, it is not able to abstract events that always occur together.

A different means of abstracting event logs is to simply remove insignificant behavior. With the fuzzy mining algorithm, Günther et al. introduce in [9], an approach to abstract a mined process model by removing and clustering less frequent behavior. In a similar manner, Polyvyanyy et al. developed in [10] a slider approach to abstract activities in given process models by using different criteria to identify insignificant activities which are to be abstracted.

Looking at the relation of different specified abstraction levels, Weidlich et al. [11] define correspondences between activities of different process models with different abstraction levels by comparing regions of activities. Yet, the approach does not consider overlapping of correspondence relations and thus, cannot handle distributed functionality as presented in this paper. Another research area which is similar to the event to activity mapping problem presented in this paper, is the research on event correlation as e.g. in [12]. The main objective of event correlation techniques is to connect events belonging to the same process instance using event attributes. Still, these approaches typically only consider 1:1 mappings. The mapping between events and activities is also similar to the matching problem known from ontologies and data integration [13].

7 Conclusion

In this paper we presented a novel approach to tackle the abstraction of event logs. Our approach distinguishes from current works by explicitly targeting a specific abstraction level and by allowing for n:m relations and concurrency.

We therefore explicitly encode domain knowledge into the mapping function in order to get the same level of abstraction as used in the defined business activities and furthermore, also allow for the specification of activity instance borders. Our approach can be used as preprocessing of event logs to lift the results of process mining techniques to a business level. We have successfully evaluated our approach and could thereby show the influence of incorrect abstractions on conformance analysis results.

Yet, our approach involves a high portion of manual work when regarding the specification of event to activity mappings. Future work should seek for possibilities to automatically propose candidate mappings, e.g. by combining our approach with the one proposed in [7]. Looking at the low level events, it should be studied how further semantics could be integrated into the abstraction approach in order to leverage life-cycle transitions such as start and end of an activity in a semi-automatic way.

References

1. van der Aalst, W.M.P.: *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, 1st edn. Springer Publishing Company, Incorporated (2011)
2. Pnueli, A.: The Temporal Logic of Programs. In: *Foundations of Computer Science*, pp. 46–57 (1977)
3. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. Morgan Kaufmann (2005)
4. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. *Information Systems* 36(7), 1009–1025 (2011)
5. Günther, C.W., van der Aalst, W.M.P.: Mining activity clusters from low-level event logs. *BETA Working Paper Series*, vol. WP 165. Eindhoven University of Technology (2006)
6. Günther, C.W., Rozinat, A., van der Aalst, W.M.P.: Activity mining by global trace segmentation. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) *BPM 2009 Workshops*. LNBP, vol. 43, pp. 128–139. Springer, Heidelberg (2010)
7. Li, J., Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Mining context-dependent and interactive business process maps using execution patterns. In: zur Muehlen, M., Su, J. (eds.) *BPM 2010 Workshops*. LNBP, vol. 66, pp. 109–121. Springer, Heidelberg (2011)
8. Greco, G., Guzzo, A., Pontieri, L.: Mining taxonomies of process models. *Data & Knowledge Engineering* 67(1), 74–102 (2008)
9. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) *BPM 2007*. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
10. Polyvyanyy, A., Smirnov, S., Weske, M.: Process Model Abstraction: A Slider Approach. In: EDOC, pp. 325–331. IEEE (2008)
11. Weidlich, M., Dijkman, R., Weske, M.: Behaviour Equivalence and Compatibility of Business Process Models with Complex Correspondences. *ComJnl* (2012)
12. Pérez-Castillo, R., Weber, B., de Guzmán, I.G.R., Piattini, M., Pinggera, J.: Assessing event correlation in non-process-aware information systems. *Software and Systems Modeling*, 1–23 (2012)
13. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer (2007)

Improving Business Process Models with Agent-Based Simulation and Process Mining

Fernando Szymanski¹, Célia G. Ralha¹, Gerd Wagner², and Diogo R. Ferreira³

¹ University of Brasília, Brazil

fszymanski@gmail.com, ghedini@cic.unb.br

² Brandenburg University of Technology, Cottbus, Germany

gwgner@informatik.tu-cottbus.de

³ IST – Technical University of Lisbon, Portugal

diogo.ferreira@ist.utl.pt

Abstract. Business processes are usually modeled at a high level of abstraction, while the analysis of their run-time behavior through process mining techniques is based on low-level events recorded in an event log. In this scenario, it is difficult to discover the relationship between the process model and the run-time behavior, and to check whether the model is actually a good representation for that behavior. In this work, we introduce an approach that is able to capture such relationship in a hierarchical model. In addition, through a combination of process mining and agent-based simulation, the approach supports the improvement of the process model so that it becomes a better representation for the behavior of agents in the process. For this purpose, the model is evaluated based on a set of metrics. We illustrate the approach in an application scenario involving a purchase process.

1 Introduction

Business processes are usually defined at a high level of abstraction using modeling languages such as BPMN [1], EPCs [2], and Petri nets [3]. In these types of models, the process is depicted as a sequence of activities, where each activity is to be performed by some agent. In human interaction workflows [4], the agent is typically a user who is able to perform certain tasks over a supporting systems infrastructure. During execution, when an agent is assigned to a certain activity, it performs a set of operations over the systems infrastructure. Among other things, such as data and information manipulation, these operations may include communicating with other agents as well.

When agents perform operations over a systems infrastructure, it is possible to record these actions in the form of events. Typically, each event refers to an operation that was performed by some agent during the execution of a process instance. Such events are recorded in an event log and, from this event log, it is possible to analyze the run-time behavior of agents through process mining techniques [5]. These techniques allow studying the run-time behavior from a number of perspectives, including the sequence of operations as well as the interactions that take place between agents during process execution.

The ultimate purpose of such analysis is to be able to compare the predefined behavior for the process with the actual behavior of agents at run-time. However, such goal faces a major obstacle. While business processes are defined and modeled at a high level, the analysis of run-time behavior through process mining techniques is based on the low-level events recorded in the event log, as each agent carries out its own operations. Clearly, there is a gap between the high level of abstraction at which processes are defined, and the low-level nature of events recorded in the event log.

The goal of this work is to bridge this gap and to provide a platform for the improvement of process models based on a combination of agent-based simulation [6–8] and process mining. Through agent-based simulation, it is possible to define, implement, and simulate a business process as a sequence of activities carried out by multiple agents working together. The interactions between these agents are recorded as low-level events. Through process mining, it is possible to extract models of behavior from this event log. However, the extraction must be done in such a way that it is possible to map low-level events to the high-level activities defined in the business process.

In summary, this work provides the following contributions:

- It shows that an agent-based simulation framework – specifically, the Agent-Object Relationship (AOR) framework [9] – can be used as a platform for implementing and simulating business processes.
- It describes a process mining technique that is able to extract a hierarchical Markov model [10] from an event log and from a description of the high-level process provided as input.
- It provides a set of metrics to evaluate the complexity of models obtained with such technique, which can serve as a guide when considering different ways to model the business process.
- It shows that the combination of both tools – i.e. the agent-based simulation framework and the proposed process mining technique – can be used as a testbed for trying out different models of the business process.

Section 2 provides an overview of the proposed approach. Section 3 discusses agent-based simulation, and in particular the use of the AOR framework in this work. Section 4 describes the hierarchical Markov model that is used to capture the run-time behavior of agents, and the algorithm to extract such model from an event log. Section 5 discusses metrics and the evaluation of the extracted models. Section 6 presents an application scenario involving in a purchase process. Finally, Section 7 concludes the paper.

2 Approach Overview

One of the premises for this work is that business experts will be able to provide a high-level description of the business process. Typically, the process is described in terms of a process model with a set of high-level activities. On the other hand, there are process mining techniques to extract the behavior of a

business process from event logs, but the low-level events that are recorded in the event log may not have a clear relationship to the high-level activities defined in the process model. Therefore, one of the main issues to be addressed is how to map the low-level events recorded in an event log to the high-level activities defined in a process model. In our approach, this is done with the aid of a process mining technique which is able to extract a hierarchical Markov model from the event log and from a Markov-model representation of the high-level business process, as shown in Figure 1.

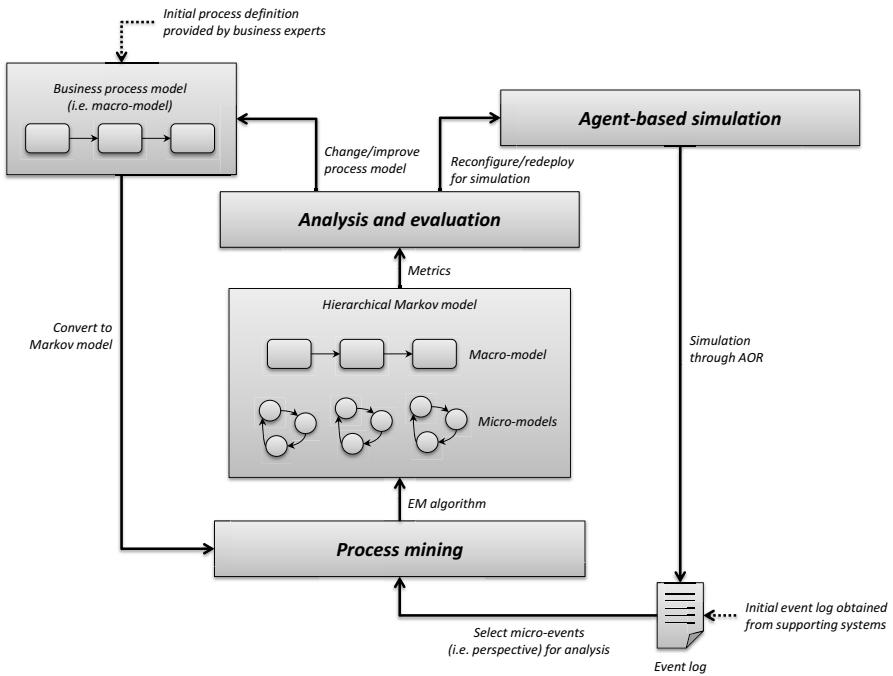


Fig. 1. Business process improvement cycle

From such hierarchical model, it is possible to assess whether the high-level process model is actually a good representation for the observed low-level behavior. In particular, it is possible to measure the quality of such model through a set of metrics. The use of metrics to evaluate the quality of process models has been thoroughly investigated in the literature [11–14], but here we focus on a set of metrics that are more tailored to the hierarchical model that is at the core of our approach. Specifically, such hierarchical model should be “balanced” in the sense that all of its components – i.e. the micro-models and the macro-model – should have a similar complexity, rather than having some components that are extremely complex and others that are oversimplified.

Following the analysis and evaluation of the hierarchical model through a set of metrics, the analyst can change the high-level description of the process in

order to create a better representation for the behavior that actually occurs in practice. However, changing the high-level process may cause a different perception of how the low-level events map to the high-level activities. For example, if an activity is split in two, or if two activities are merged together, the relationship between the low-level events and the high-level activities will change. Contrary to what may appear at first sight, such change is not entirely predictable, since agents organize themselves in a non-deterministic way to carry out the process. In addition, process mining techniques do not provide perfect accuracy, so one can gain further insight into the run-time behavior of the process by trying out different configurations for the process model.

Ideally, a new version of the process would be deployed in the organization and one would be able to collect new event logs. Then one could run the process mining algorithm again in order to check which changes have been introduced in the run-time behavior, and whether the new process is a good representation of that behavior. In practice, one may not be able to do such experiments on the real-world environment due to the risks, costs or time involved. Therefore, we introduce the possibility of carrying out an agent-based simulation for the new process model. This simulation will be configured with the knowledge that has been collected so far about the process.

In previous work [10], we have shown that it is possible to generate event logs from agent-based simulations. In particular, we used the AOR framework [9] for that purpose. With this platform, it is possible to configure simulation scenarios which comprise multiple agents interacting with each other. An event log of these interactions can be recorded and used for process mining analysis. Together with the high-level description of the business process, this event log can be used to extract a new hierarchical model and again analyze and evaluate this model in order to assess the opportunity for further changes. This improvement cycle is based on simulation and mining, and it can be repeated until a satisfactory process model is found.

3 Agent-Based Simulation

Agent-based simulation (ABS) [6–8] focuses on the analysis of business systems involving interactions among agents. ABS can be used to represent organizational settings in a natural way, since they involve business actors that communicate and interact with each other. In particular, it is possible to use ABS for simulating the execution of a business process, thereby generating an event log. In our approach, we use the AOR simulation framework [9], which provides both high-level constructs (such as activities) and low-level constructs (such as incoming and outgoing message events) to facilitate the mapping of a business process model into a simulation model.

In the AOR framework, agents react to events in their environment by performing actions and by interacting with each other. There are basically two different kinds of events:

- An *exogenous event* is an external event (such as the arrival of a new customer) which is not caused by any previous event. Usually, the occurrence of such an event triggers a new instance of the business process. To run multiple instances of a business process, the AOR system schedules several exogenous events to trigger the process at different points in time.
- The second kind of event is a *caused event*. For example, if agent X sends a message M1 to agent Y, then this may result in another message M2 being sent from agent Y to agent Z. Agents send messages to one another for different purposes, e.g. for reporting, for making requests and for responding. The chaining of messages through caused events is what keeps the simulation running until there are no more events.

The specification of a simulation scenario begins by defining a set of *entity types*, including different types of agents, messages and events. The behavior of agents is specified by means of *reaction rules*. Typically, such a rule defines that when a certain message is received, the information state of the agent is updated and another message is sent to some other agent. Since the rules for each agent are defined separately, the simulation scenario is effectively implemented in a decentralized way by the combined behavior of all agents.

A second kind of rule in an AOR simulation scenario are *environment rules*. While reaction rules define the behavior of agents, environment rules define the behavior associated with the external environment. An environment rule specifies that when an exogenous event occurs, the state of certain objects is changed and certain follow-up events result from it. For example, an environment rule may specify that when a certain event occurs, a message is sent to an agent; sending this message then triggers a reaction rule of the receiving agent, creating a chain of events that puts the simulation in motion.

Environment rules also have the ability to create (or destroy) agents. This is especially useful to simulate, for example, the arrival (or departure) of customers. A set of *initial conditions* for the simulation scenario specifies which agents already exist in the scenario at the beginning of the simulation. The initial conditions also include a schedule for the occurrence of at least one exogenous event to trigger the simulation.

All of these constructs (i.e. entity types, reaction rules, environment rules, and initial conditions) are specified using an XML-based language called *AOR Simulation Language* (AORSL) [15]. The specification of a simulation scenario in AORSL is transformed into Java code by the AOR system. Running the simulation amounts to running this auto-generated Java code.

4 Process Mining with Hierarchical Markov Models

Process mining [5] includes a number of different perspectives, namely the control-flow perspective, the organizational perspective, and the performance perspective. In each of these perspectives there are a number of specialized techniques, such as the α -algorithm [16] for the control-flow perspective, the social

network miner [17] for the organizational perspective, or the dotted chart [18] for the performance perspective. However, these techniques have in common the fact that they work at the same level of abstraction as the events recorded in the event log, so they may not be very helpful in providing insight in terms of the high-level activities that are used to describe the business processes.

More recently, there has been some effort in developing techniques that are able to address this problem. Some of these techniques work by extracting a low-level model from the event log and then creating a more abstract representation of that model [19, 20]. Other techniques begin by translating the event log into a more abstract sequence of events, and then extracting models from that translated event log [21, 22]. Here we address the problem with a special kind of model – i.e. the hierarchical Markov model introduced in [10] – which is able to capture both the high level of abstraction at which the process is defined and the low-level behavior that can be observed in the event log.

Figure 2 illustrates a simple example of a hierarchical Markov model. Here, the business process is described on a high level as comprising the sequence of activities A, B, and C. When each of these activities is performed, this results in some sequence of low-level events being recorded in the event log. Figure 2 represents these low-level events as X, Y and Z. In practice, these may represent the actions that are being carried out by agents, or they may also represent the interactions, i.e. the messages exchanged between agents. The meaning of X, Y and Z depends on the type of events that are recorded in the event log.

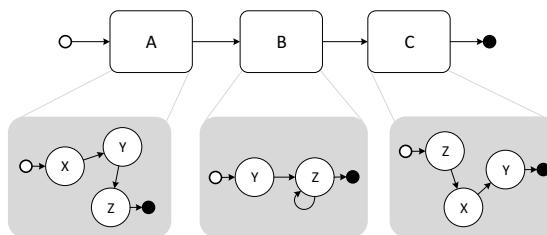


Fig. 2. A simple hierarchical Markov model

The hierarchical model in Figure 2 means that executing activity A results in the sequence of events XYZ being recorded in the event log. In a similar way, activity B results in a sequence of events in the form YZZ..., where there may be multiple Z's until a certain condition becomes true. Finally, activity C results in a sequence of events in the form ZXY. Both the high-level process and these sequences of low-level events are represented as Markov chains. Executing this model corresponds to performing the sequence of activities ABC. However, in the event log we find sequences of events such as XYZYZZZXY.

The sequence ABC is called the *macro-sequence*, and the high-level Markov chain that represents the business process expressed in terms of the activities A, B, and C is referred to as the *macro-model*. On the other hand, the sequence of

events XYXYZZZXY is called a *micro-sequence*, and the low-level Markov chains that describe the behavior of each macro-activity in terms of the events X, Y and Z are referred to as the *micro-models*.

The hierarchical Markov model relates to the approach depicted in Figure 1 in the following way: the high-level description of the business process corresponds to the macro-model, and the low-level event log corresponds to the micro-sequence.¹ Therefore, both the macro-model and the micro-sequence are known. The problem is how to discover the macro-sequence and the micro-models from the given macro-model and micro-sequence.

This problem has been addressed in detail in [10], and it can be solved with an Expectation-Maximization (EM) procedure [23] as follows:

1. Draw one macro-sequence at random from the macro-model (in the example of Figure 2 only one macro-sequence is possible: ABC).
2. From the macro-sequence and the micro-sequence, find an estimate for the micro-models (Algorithm 2 in [10]).
3. From the macro-model and the micro-models from step 2, find the most likely macro-sequence (Algorithm 3 in [10]).
4. With the macro-sequence from step 3, repeat step 2. Then with the micro-models from step 2, repeat step 3. Do this until the micro-models converge.

In [24] the authors show that this EM procedure is able to deal with workflow patterns such as branching, parallelism and loops. Our goal here is just to highlight that from a high-level model of the business process (i.e. the macro-model) and a low-level event log recorded during execution (i.e. the micro-sequence), it is possible to derive a set of micro-models that capture the low-level behavior within each high-level activity, as depicted in Figure 2. The initial macro-model, together with the recently discovered micro-models, can then be evaluated through a set of metrics, as explained in the next section.

5 Evaluation Metrics

In the literature, there are several metrics that have been proposed for the analysis of business process models [11–14]. In some cases, these metrics have their origin in network analysis and software engineering.

Metrics for describing network structure are usually derived from graph theory and network theory. For example, there are some characteristics that are normally used to describe a network structure, e.g. centrality, degree, density, and connectivity [25]. A business process model, expressed in BPMN for example, can be viewed as a special kind of graph, and therefore those network analysis techniques can be applied in this context as well.

In software engineering, there are also several metrics to measure aspects that are relevant for quality assurance. These metrics depend on the programming

¹ More precisely, an event log usually contains multiple traces, and each trace corresponds to a separate micro-sequence.

paradigm being used. In procedural programming, the most common metrics are structural and they are based on the counting of functions along the control-flow, such as the cyclomatic number [26], the information flow metric (fan-in, fan-out) [27], and the COnstructive COst MOdel [28], among others. In object-oriented programming, there is a different set of metrics to measure factors such as cohesion and coupling [29]. To some extent, some of these metrics can be used to evaluate business process models too.

In this work, we are interested in metrics that can be used to evaluate process models expressed as hierarchical Markov models. For this purpose, we studied a number of metrics focusing on factors such as size, density, modularity, control-flow, etc. [11, 30, 31]. The literature provides a lot of metrics to measure these factors, but here we selected a subset of metrics that are more geared towards the structure and complexity of those models. In particular, we are interested in metrics that allow us to determine if the hierarchical model is “balanced” in the sense that the macro-model and the micro-models should have similar complexity. The chosen metrics are summarized in Table 1.

Table 1. Metrics to evaluate the components of a hierarchical model

Metric	Abbrev.	Focus	Definition
No. of arcs per node [11]	NAN	Density	Ratio of number of arcs to number of nodes.
Relational density [32]	RD	Density	Ratio of number of arcs to the total number of possible arcs.
No. of paths [26]	NP	Control-flow	Number of all possible distinct paths between start node and end node.
Path length [25]	PL	Size	Number of nodes between start node and end node. (For a given model, this is calculated as the average length of all possible paths.)
Cyclomatic complexity [26]	CC	Control-flow	Number of linearly-independent paths. Can be measured as $M = E - N + 2P$ where E is the number of arcs, N is the number of nodes, and P is the number of exit nodes.
Fan-in/Fan-out [14]	FIO	Modularity	$(f_{in} \cdot f_{out})^2$ where f_{in} is the number of nodes that precede a given node, and f_{out} is the number of nodes that follow a given node. (For a given model, f_{in} and f_{out} are calculated separately as an average across all nodes.)
No. of sub-processes	SUB	Modularity	Number of nested sub-processes in a model.

All metrics (except SUB) can be calculated separately for the macro-model and for each micro-model. This provides an assessment of each component in the hierarchical model. For example, in the hierarchical model of Figure 2 there are four components, and the metrics can be calculated for each of them. To get an assessment of the hierarchical model as a whole, in this work we take a simple average of the results obtained across all components.

6 Application Scenario

The purpose of this application scenario is to illustrate how the proposed approach can be used to facilitate the refinement of a process model, so that this model becomes not only a more precise representation of the observed behavior, but also a model which achieves better results in terms of the metrics described in the previous section. The scenario involves a purchase process which is described on a high-level both in text and by means of a BPMN diagram. Due to space restrictions, we will show the initial version (version 1) and the final version (version 2), but there could be other versions in between.

Initially, the process is described as follows:

In a company, an employee needs a certain commodity (e.g. a printer cartridge) and submits a request for that product to the warehouse. If the product is available at the warehouse, then the warehouse dispatches the product to the employee. Otherwise, the purchasing department buys the product, and then the warehouse dispatches the product to the employee.

This process is represented as a BPMN diagram in Figure 3, and its control flow can be expressed as a Markov chain with transition probabilities, as shown in Figure 4. In this macro-model, there are just three high-level activities and one decision. For lack of further info, we assume that the two possible outcomes from this decision are equally likely, so the transition probabilities from the “Requisition” activity to “Dispatch Product” and “Buy Product” are both 0.5 in Figure 4. In this Markov chain, there are also two special states (\circ and \bullet) to represent the beginning and end of the process, respectively.

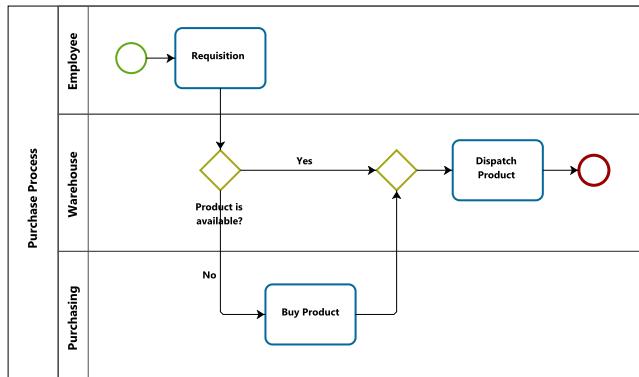


Fig. 3. BPMN diagram for the purchase process (version 1)

In a real scenario, an initial event log can be obtained from the supporting systems in order to capture the low-level operations performed by the participants in the process, or the interactions (i.e. message exchanges) between those participants. Table 2 shows an example of the later. Here, each message has



Fig. 4. Markov chain representation for the purchase process (version 1)

been represented as having a certain type or meaning. For example, StockRequest is a message sent from the Employee to the Warehouse. While this message appears to be related to the “Requisition” activity, for other messages the relationship to a high-level activity may not be so clear.

Table 2. Excerpt of an event log

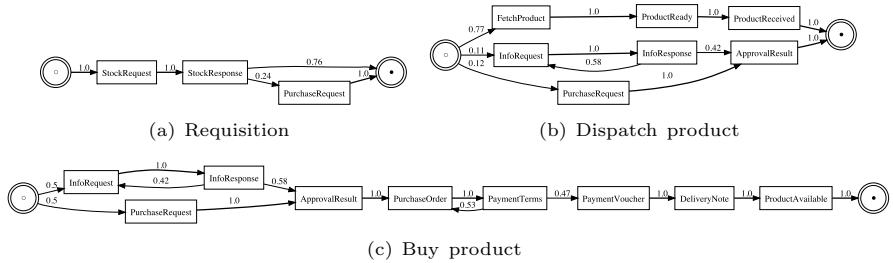
case_id	sender	message	receiver	timestamp
1	Employee	StockRequest	Warehouse	2013-02-02 11:26
1	Warehouse	StockResponse	Employee	2013-02-04 16:07
1	Employee	FetchProduct	Warehouse	2013-02-05 08:54
1	Warehouse	ProductReady	Employee	2013-02-06 10:23
1	Employee	ProductReceived	Warehouse	2013-02-07 15:47
2	Employee	StockRequest	Warehouse	2013-02-12 09:31
2	Warehouse	StockResponse	Employee	2013-02-14 14:10
2	Employee	PurchaseRequest	Purchasing	2013-02-15 16:35
2	Purchasing	InfoRequest	Employee	2013-02-18 17:21
2	Employee	InfoResponse	Purchasing	2013-02-19 10:52
2	Purchasing	ApprovalResult	Employee	2013-02-20 12:05
2	Purchasing	PurchaseOrder	Supplier	2013-02-21 15:19
...

From the event log in Table 2 it is possible to retrieve the sequence of events (i.e. micro-sequence) for each process instance (i.e. case id). Also it is possible to select one of the columns *sender*, *message* and *receiver* for analysis. Here we will use the *message* column, so the micro-sequence for case 1 is:

StockRequest → StockResponse → FetchProduct → ProductReady → ProductReceived

The micro-sequences for other cases can be extracted in a similar way. These micro-sequences, together with the macro-model in Figure 4, are provided as input to the algorithm described in Section 4, which produces the micro-models shown in Figure 5. Note that in Figure 5 there is some duplicated behavior in the “Dispatch product” and “Buy product” activities, resulting in longer and more complex micro-models. These micro-models were evaluated using the metrics defined in Section 5 and the results are shown in Table 3.

The results in Table 3 reflect the fact that the micro-models in Figure 5 are relatively more complex when compared to the macro-model in Figure 4. In particular, the metrics NP (no. of paths) and PL (path length) are significantly higher for the micro-models when compared to the same metrics for the macro-model. This means that the hierarchical model is somewhat “unbalanced”, in the sense that the macro-model is of significant less complexity than the micro-models, and therefore it is probably an over-simplified representation of the

**Fig. 5.** Micro-models for the purchase process (version 1)**Table 3.** Metrics applied to the purchase process (version 1)

Model	Metrics						
	NAN	RD	NP	PL	CC	FIO	SUB
Macro model	1.00	0.33	2.00	2.50	2.00	3.13	–
Micro Model (Requisition)	1.00	0.33	2.00	2.50	2.00	1.77	–
Micro Model (Dispatch Product)	1.22	0.17	4.00	3.25	4.00	2.16	–
Micro Model (Buy Product)	1.18	0.13	6.00	9.33	4.00	2.63	–
Complete model (avg.)	1.10	0.24	3.25	4.40	3.00	2.42	3

business process. There appears to be much more behavior in the micro-models than what the macro-model is able to account for.

Therefore, the business experts are encouraged to describe the process in more detail. Such description could be as follows:

In a company, an employee needs a certain commodity (e.g. a printer cartridge) and submits a request for that product to the warehouse. If the product is available at the warehouse, then the warehouse dispatches the product to the employee. Otherwise, the product must be purchased from an external supplier. All purchases must be approved by the purchasing department. If the purchase is not approved, the process ends at that point. On the other hand, if the purchase is approved, the purchasing department orders and pays for the product from the supplier. The supplier delivers the product to the warehouse, and the warehouse dispatches the product to the employee.

This new version of the process is depicted in Figure 6 and is expressed as a Markov chain (i.e. macro-model) in Figure 7. On the other hand, to mine the hierarchical model, we will need an event log with the run-time behavior for this process. Rather than deploying the model in the organization and waiting for an event log to be recorded over a long period of time, we use the AOR framework to build an agent-based simulation scenario based on the previous version of the process (i.e. the hierarchical model in Figures 4 and 5).

If the behavior of the high-level process had been changed in this new version (e.g. if two high-level activities had been switched in their execution order) then we would need to adapt the simulation scenario in order to reflect those changes. However, here the new macro-model is just a refinement of the previous one, so

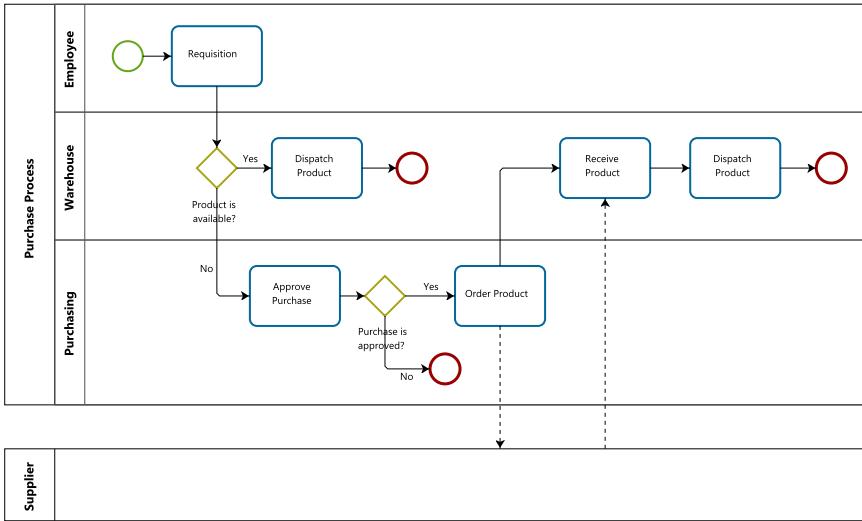


Fig. 6. High-level description of the purchase process (version 2)

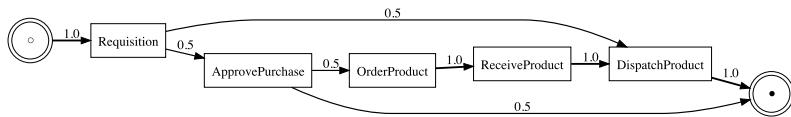


Fig. 7. Markov chain representation for the purchase process (version 2)

we can use the previous hierarchical model as a basis for simulation, without the need for further changes. Even though the hierarchical model in Figures 4 and 5 is not a balanced model, for simulation purposes it is still a perfectly valid model for reproducing the behavior of the process.

By running the simulation scenario in AOR, it is possible to collect a new and possibly larger event log to mine the next version of the hierarchical model. The event logs that are generated by the AOR framework are in XML, but they can be easily converted to the tabular form of Table 2. In this experiment, we ran a simulation with 10,000 steps, which produced an event log with 140 process instances and a total of 1136 events. The event log, together with the macro-model of Figure 7, were provided as input to the algorithm of Section 4, which produced the micro-models shown in Figure 8.

This new hierarchical model (i.e. Figures 7 and 8) was evaluated using the same metrics as before. As can be seen in Table 4, this new model is more balanced because the micro-models are, in general, of significantly lower complexity than before, while the macro-model is only slightly more complex. Comparing the last row of Table 4 with the last row of Table 3 shows that, overall, the new model is less complex than the previous one. In particular, NP (no. of paths),

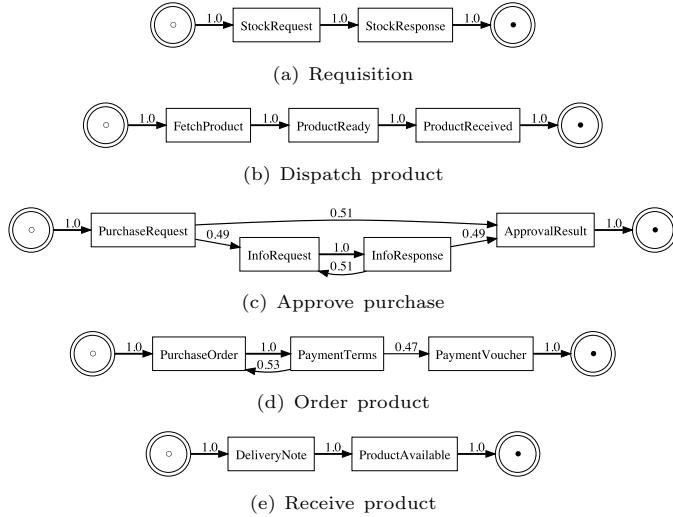


Fig. 8. Micro-models for the purchase process (version 2)

Table 4. Metrics applied to the purchase process (version 2)

Model	Metrics						
	NAN	RD	NP	PL	CC	FIO	SUB
Macro model	1.14	0.23	3.00	3.00	3.00	2.82	—
Micro Model (Requisition)	0.75	0.38	1.00	2.00	1.00	1.00	—
Micro Model (Dispatch Product)	0.80	0.27	1.00	3.00	1.00	1.00	—
Micro Model (Approve Purchase)	1.17	0.29	3.00	4.00	3.00	5.06	—
Micro Model (Order Product)	1.00	0.33	2.00	4.00	2.00	3.13	—
Micro Model (Receive Product)	0.75	0.38	1.00	2.00	1.00	1.00	—
Complete model (avg.)	0.93	0.31	1.83	3.00	1.83	2.34	5

PL (average path length), and CC (cyclomatic complexity) are significantly lower than before. This means that the model in Figure 6 is not only a more accurate description of the process, but also the low-level behavior associated with each high-level activity is simpler and easier to understand.

7 Conclusion

In this work we described an iterative approach for the improvement of business process models based on process mining and agent-based simulation. The need for such approach is justified by the fact that there is a gap between the high-level of abstraction at which processes are usually modeled and the low-level nature of events that are generated during execution. The process mining technique that we used here is able to capture this relationship in the form of a hierarchical Markov model. On the other hand, an agent-based simulation platform is used as a means to generate the low-level behavior for new versions of

the process. Provided with a set of metrics that serve as guidance, the process analyst can insert changes to the process model, reconfigure the simulation platform, generate a new event log, and mine a new hierarchical model that again captures the relationship between the high-level activities and the low-level behavior. Doing this iteratively will lead to a better model, where “better” means more accurate, more balanced, less complex, and easier to understand.

Due to space restrictions, here we focused on the analysis of the control-flow alone, but the same approach can be applied to the analysis of the organizational perspective, which includes the handover of work between agents and the collaboration of agents within each case. This can be done by selecting other columns for analysis, namely the *sender* column or the *receiver* column in the event log of Table 2. In future work, we are planning to improve several aspects of the proposed approach, namely establishing guidelines for the conversion of BPMN models to Markov models, supporting the automatic generation of an AOR simulation scenario from a given hierarchical model, and expanding the set of metrics used in the analysis and evaluation phase.

References

1. OMG: Business Process Model and Notation (BPMN), Version 2.0 (2011)
2. Scheer, A.W.: ARIS: Business Process Modeling, 3rd edn. Springer (2000)
3. van der Aalst, W.M.P.: The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers* 8(1), 21–66 (1998)
4. Weske, M.: Business Process Management: Concepts, Languages, Architectures. 2nd edn. Springer (2012)
5. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
6. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. *PNAS* 99(suppl. 3), 7280–7287 (2002)
7. Davidsson, P., Holmgren, J., Kyhlbäck, H., Mengistu, D., Persson, M.: Applications of agent based simulation. In: Antunes, L., Takadama, K. (eds.) MABS 2006. LNCS (LNAI), vol. 4442, pp. 15–27. Springer, Heidelberg (2007)
- 8.Railsback, S.F., Lytinen, S.L., Jackson, S.K.: Agent-based simulation platforms: Review and development recommendations. *Simulation* 82(9), 609–623 (2006)
9. Wagner, G.: AOR modelling and simulation: Towards a general architecture for agent-based discrete event simulation. In: Giorgini, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS 2003. LNCS (LNAI), vol. 3030, pp. 174–188. Springer, Heidelberg (2004)
10. Ferreira, D.R., Szimanski, F., Ralha, C.G.: A hierarchical Markov model to understand the behaviour of agents in business processes. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBP, vol. 132, pp. 150–161. Springer, Heidelberg (2013)
11. Mendling, J.: Metrics for Process Models. LNBP, vol. 6. Springer, Heidelberg (2009)
12. Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Quality Metrics for Business Process Models. In: 2007 BPM & Workflow Handbook, pp. 179–190. Future Strategies Inc. (2007)
13. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. *Information Systems* 36(2), 498–516 (2011)

14. Gruhn, V., Laue, R.: Approaches for Business Process Model Complexity Metrics. In: Technologies for Business Information Systems, pp. 13–24. Springer (2007)
15. Nicolae, O., Wagner, G., Werner, J.: Towards an executable semantics for activities using discrete event simulation. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009 Workshops. LNBP, vol. 43, pp. 369–380. Springer, Heidelberg (2010)
16. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering* 16, 1128–1142 (2004)
17. Song, M., van der Aalst, W.M.P.: Towards comprehensive support for organizational mining. *Decision Support Systems* 46(1), 300–317 (2008)
18. Song, M., van der Aalst, W.M.P.: Supporting process mining by showing events at a glance. In: Proceedings of 17th Annual Workshop on Information Technologies and Systems, pp. 139–145 (2007)
19. Greco, G., Guzzo, A., Pontieri, L.: Mining hierarchies of models: From abstract views to concrete specifications. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 32–47. Springer, Heidelberg (2005)
20. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
21. Günther, C.W., Rozinat, A., van der Aalst, W.M.P.: Activity mining by global trace segmentation. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009 Workshops. LNBP, vol. 43, pp. 128–139. Springer, Heidelberg (2010)
22. Bose, R.P.J.C., Verbeek, E.H.M.W., van der Aalst, W.M.P.: Discovering hierarchical process models using ProM. In: Nurcan, S. (ed.) CAiSE Forum 2011. LNBP, vol. 107, pp. 33–48. Springer, Heidelberg (2012)
23. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions. Wiley Series in Probability and Statistics. Wiley-Interscience (2008)
24. Ferreira, D.R., Szymanski, F., Ralha, C.G.: Mining the low-level behavior of agents in high-level business processes. *International Journal of Business Process Integration and Management* (to appear, 2013)
25. Newman, M.: The structure and function of complex networks. *SIAM Review* 45(2), 167–256 (2003)
26. McCabe, T.J.: A complexity measure. *IEEE Transactions on Software Engineering* 2(4), 308–320 (1976)
27. Henry, S.M., Kafura, D.G.: Software structure metrics based on information flow. *IEEE Transactions on Software Engineering* 7(5), 510–518 (1981)
28. Boehm, B.W.: Software engineering economics. Prentice-Hall (1981)
29. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. *IEEE Transactions on Software Engineering* 20(6), 476–493 (1994)
30. Nissen, M.E.: Redesigning reengineering through measurement-driven inference. *MIS Quarterly* 22(4), 509–534 (1998)
31. Cardoso, J.: Control-flow complexity measurement of processes and Weyuker's properties. In: 6th International Enformatika Conference. Transactions on Enformatika, Systems Sciences and Engineering, vol. 8, pp. 213–218 (October 2005)
32. Vanderfeesten, I.T.P., Reijers, H.A., Mendling, J., van der Aalst, W.M.P., Cardoso, J.: On a quest for good process models: The cross-connectivity metric. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 480–494. Springer, Heidelberg (2008)

Towards a Framework for Modeling Business Compensation Processes

Anis Boubaker¹, Hafedh Mili¹, Yasmine Charif², and Abderrahmane Leshob¹

¹ University of Quebec at Montreal (UQÀM), LATECE Laboratory

² Xerox Research Center at Webster

Abstract. A typical e-business transaction takes hours or days to complete, involves a number of partners, and comprises many failure points[8]. With short-lived transactions, database systems ensure atomicity by either committing all of the elements of the transaction, or by canceling all of them in case of a failure. With typical e-business transactions, strict atomicity is not practical, and we need a way of reversing the effects of those activities that cannot be rolled back: that is *compensation*. For a given business process, identifying the various failure points, and designing the appropriate compensation processes represents the bulk of process design effort[8]. Yet, business analysts have little or no guidance, as for a given failure point, there appears to be an infinite variety of ways to compensate for it. We recognize that compensation is a business issue, but we argue that it can be explained in terms of a handful of parameters within the context of REA ontology [20], including things such as the type of activity, the type of resource, and organizational policies. We propose a three-step process compensation design approach that 1) starts by abstracting a business process to focus on those activities that create/modify value, 2) compensates for those activities, individually, based on values of the compensation parameters, and 3) composes those compensations using a Saga-like approach [10]. In this paper, we present our approach, and discuss issues for future research.

1 Introduction

Consider an order and delivery process used by an e-retailer such as *Amazon*. The process starts when the customer gets online and places an order. The ordered items (say, books) are then checked for availability by the *Amazon* warehouse. At the same time, the customer's payment information is ran through a verification process by a financial institution, releasing a debit authorization if it succeeds. In case debit authorization fails, the process aborts and the order is canceled. Otherwise, the order is packaged and shipped to the customer, using transportation services provided by a shipping company. In the meantime, the order amount is charged to customer's credit card by the financial institution. The process ends when the customer takes possession of the goods she ordered.

This process description establishes the "*happy path*". However, every robust business process must address alternate paths handling situations where something could go wrong. In our example, we could envision situations where, say,

the book is out of stock, or the payment is rejected by the customer's financial institution, or the wrong book is shipped, or the order is delivered to a wrong address or, more simply, the order was cancelled by the customer at any moment within the process.

A typical e-business transaction takes hours or days to complete, involves a number of partners, and may comprise a great number of failure points[8]. Each failure point may involve undoing some steps (pretending they never happened) or reversing their effects fully or partially. Database research has thoroughly addressed the problems raised by long running transactions (LRT), aiming to achieve relaxed atomicity to the global transaction by ensuring that either the process completes successfully as a whole or to have its effects reversed (e.g. [10,5]). Due to the long running nature of business processes, it is unthinkable to lock the resources to ensure ACID properties. Approaches like Sagas[10] consist of slicing the business process into a set of activities treated as ACID transactions (i.e. Sagas). If one Saga fails at runtime, then the whole process should stop and the running Saga should be treated by a regular rollback. However, previously committed Sagas cannot be rolled back [21] and their semantic effects must be reversed in order to preserve system's consistency. This is what is called a compensation process. As stated in the BPMN standard, compensation is concerned with undoing steps that were already successfully completed, because their results and possibly side effects are no longer desired and need to be reversed[23].

Some studies report that nearly 80% of the time spent on implementing a business process is spent on handling possible exceptions/failures (as mentioned in [18]). An overriding issue seems to be that there appears to be numerous ways of compensating for a single activity, and business analysts and process designers, alike, are left with no assistance, and few guidelines, if any, to design compensation activities. To make problems worse, process designers are often expected to figure out how to compensate for activities taking place within business partners. Our objective is to develop tools and techniques to help business analysts design compensation activities.

We recognize that compensation is primarily a business issue. However, that does not mean that it cannot be explained or rationalized. To the contrary, we argue that the major business decisions that underlie a compensation process can be explained in terms of a handful of parameters that capture the (business) essence of the products and services being manipulated by the process, and the nature of those manipulations. To get to this level of analysis, we need to abstract away from the low-level implementation details of the process (e.g. the interfaces/APIs of the services invoked), and focus on the underlying *business transactions*.

Many authors argued that the essence of a business process resides in the creation of *value* by consuming or transforming a set of *resources* in order to obtain another set of resources perceived by the customer as having a greater overall value (e.g. [16]). We share this view and we believe that a value-based process modeling is the right level of abstraction for representing the business decisions that

underlie compensation. More specifically, a *resource-event-agent* (REA)-based value modeling [11], which focuses on the resources exchanged or transformed during a business activity, provides a useful metaphor to think about compensation. In particular, within the context of REA, we have been able to identify *seven* parameters whose values determine compensation. We propose a three-step process compensation design approach that 1) starts by abstracting a business process to focus on those activities that create/modify value, 2) compensates for those activities, individually, based on values of the compensation parameters, and 3) composes those compensations using a Saga-like approach [10].

The remainder of this paper goes as follows. Section 2 briefly describes the REA framework. In section 3 we present the compensation decision factors and illustrate our approach through our *Amazon* running example. We then relate our work to previous research and discuss outstanding issues in section 4, before concluding in section 5.

2 The REA Framework

In an early work, McCarthy[20] introduced the Resource-Event-Agent (REA) framework as an accounting framework aiming to record economic phenomena in a shared data environment. It has since been used as an approach to conceptualize and record business activities within an information system and its foundation as a business ontology has been established (see [12,13]). Essentially, the REA framework enables us to model business activities in pure business terms using a small set of concepts and the relationships between them.

To illustrate the main concepts of the REA framework, let's consider the *Amazon* example introduced earlier. Taken from a high level, the entire process is concerned with providing a customer with a book in exchange for a money payment (see Fig. 1). Both *Amazon* and the customer undertake actions in order to achieve this business objective. Intuitively, the REA framework permits us to model the business process in terms of business assets - i.e. the *Resources*- that are controlled by process participants - i.e. the *Agents* - and *exchanged* within economic *events*. In the following, we describe each of these concepts.

A business process is a set of activities carried on in order to achieve some business objective. These activities utilize valuable assets as raw material, employee labor, money, and so on. We call these assets *Economic Resources*. McCarthy defines them as being “objects that (1) are scarce and have utility and (2) are under the control of an enterprise” [20].

Activities of the business process are performed by physical persons (e.g. our customer) or moral entities - i.e. organizations or organizational units - (e.g. *Amazon*) called *Economic Agents*. Agents have control over the involved economic resources and are granted the ability to relinquish or acquire them. We associate each economic resource to two economic agents: the one who *provides* the economic resource and the one who *receives* it.

Finally, the REA framework conjectures that each economic phenomenon involves complementary activities called *economic events*: one that increases some

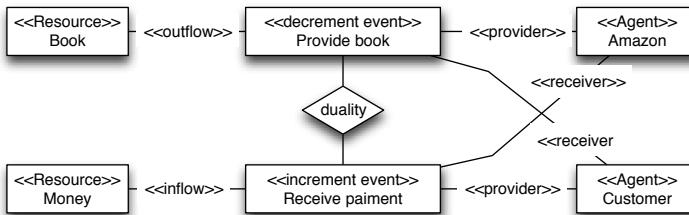


Fig. 1. A high level REA model of *Amazon's* process

resources on hand - the *increment event* - and one that relinquishes some other resources under the control of the company - the *decrement event*. McCarthy adopted Yu's definition of an economic event[28] as being "a class of phenomena which reflect changes in scarce means resulting from production, exchange, consumption and distribution". Increment and decrement events are linked to their corresponding resources through a *stock-flow* association - i.e. *inflow* and *outflow*, respectively.

Increment and decrement events of a given economic phenomenon are also linked through a *duality* relationship exhibiting the *exchange* between involved agents. Thus, the exchange shows the essence of value creation and aims to rationalize the business process by exhibiting why resources have been relinquished. Note that REA events differ from the traditional metaphor of events as seen in programming languages (e.g.: WS-BPEL) by being long lasting rather than instantaneous.

The discussion above introduces two aspects the REA ontology permits us to represent. First, REA enables us to exhibit the transactional nature of a business process by exposing the sequence of what is relinquished by the process and what is gained in exchange. Thus the concept of an exchange is the cornerstone of the REA ontology and establishes the rationale behind business activities by recording *why* the business engages in such activities. Second, the REA framework aims at exhibiting *how* value is created through business activities by modeling a value chain as introduced by Porter[25].

In the next section we describe our approach to modeling compensation processes relying on the REA abstraction.

3 Value-Oriented Compensation - Our Approach

We argued in[21] that current implementations of compensation in service-oriented architectures have inherent problems in regards to language constructs. The designer of a service orchestration (i.e. the consumer of web services) has the responsibility to account for the many exceptions errors that may occur during the execution of a business process with little or no guidance. In other words, the designer has to know what services he should invoke to compensate a given

service or has to implement his own compensation activities – thus impeding service reusability.

Notwithstanding the technical issues, we share the view that compensation is first and foremost a business problem. However, we argue that behind the seemingly infinite variety of compensation responses that organizations can deploy to a given failure, there lay a handful of principles that we should be able to codify [21]. This, in turn, relies on our ability to analyze the activities within a typical business process in pure economic and business terms, abstracting away—for the time being—the idiosyncrasies of the corresponding record-keeping by the IT system. To this end, we propose, as a first step to our approach, to use the REA ontology for analyzing business processes. The second step defines the requirements of compensation activities relying on a set of factors expressed in pure business terms. The last step will construct the compensation process value chain intended to provide the process designer with a desirable insight on which compensation activities should be considered and what each activity should be concerned about.

In the following subsections, we will build on the *Amazon* example introduced earlier in order to explain each step of our approach.

3.1 REA Value-Chain Design

In Figure 1 we have modeled our *Amazon* process at a level that does not account for all the resources that are consumed in the process of delivering the book to the customer— and that may need to be compensated for in the case of a process failure. For example, the exchange of book versus money could not happen as depicted in Fig. 1 since both parties are in different locations. Indeed, the book needs to be transported to customer’s location and the money has to be somehow collected by *Amazon*. In order to change the *Book*’s location property, *Amazon* relies on a *shipping company* who provides it with transportation rights. Similarly, the change of *Money*’s location involves a *financial institution* handling the credit card transaction, which, in turn, involves transaction fees charged by the *financial institution*.

The REA framework allows the description of the business process at different levels of abstraction. It is theoretically possible to develop a model that encompasses all economic phenomena. However, this needs to be balanced against the need to keep our models manageable and scalable. In [11], Geerts et al. suggest to decompose business processes down to the level needed "to plan, control and evaluate" the business process

In our example, for the sake of simplicity, we suppose that the transportation service has been previously acquired from the shipper (see E2, Fig. 2). This service is consumed to ship the book, resulting in the E3 REA conversational exchange, shown on Fig. 2. As for the banking transaction fees, we made the (modeling) decision to not account for them as a full fledged resources, along with the corresponding events and duality relationship. Thus, we chose to represent them as an attribute of the *Receive Payment* event (E4). By continuing further down we notice that the book cannot be shipped as is and must be packaged

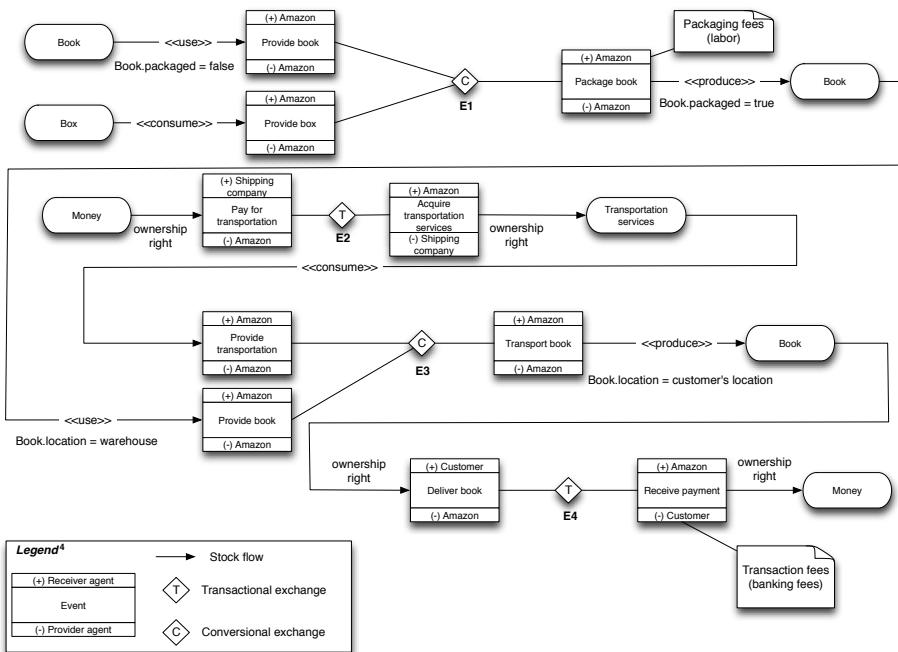


Fig. 2. REA value chain of *Amazon's* process

prior to shipping. This transformation requires the consumption of a box as well as the labor of an employee (E1). As we did with banking fees, we decide to account for employees' labor as an attribute of the packaging activity.

Once we have determined the REA exchanges at the desired level of abstraction, we construct the global value chain by connecting the exchanges. Fig. 2 depicts the global value chain for our *Amazon* example. Section 3.2 explains the compensation decision factors needed to compensate for the various exchanges.

3.2 Compensation Factors

Consider the exchange E1 from Fig. 2, which creates a "book in a box" from a book and a packing service. If the process aborts after E1 has been performed, we need to figure out how to reverse this exchange. This, in turn, requires us to understand, among other things, what activities have taken place, and how they affect the manipulated resources: (1) a box has been consumed and is no longer usable for future exchanges, (2) the book is now enclosed in a box, and (3) labor has been consumed— in this case, represented as an attribute of E1. Intuitively, we can return the book to its previous state, by spending a bit more labor. However, the box is "lost forever". This illustrates two compensation decision parameters, 1) the *type of process*, and 2) the way it affects its input resources

(the book versus the box). Luckily, these parameters have a handful of values each: 1) an exchange can be either *conversional* or *transactional*, and 2) the effect of a conversional exchange on a resource can be either *reversible* or not.

We have identified seven compensation decision factors, which fall into two groups, 1) what we called *class factors*, that depend on things such as *types* of (REA) events and resources, and that determine *whether* an exchange or resource needs to be compensated, and *how*, and 2) *instance factors*, which determine, for a particular exchange *instance, whom* and *how much* to compensate. We present the two groups in turn.

Class Factors

What is the Type of the Exchange? In order to compensate for a given activity, one should be able to asses the effects of the activity on the economic resources. The REA framework distinguishes two types of exchanges: *transactional exchanges* and *conversional exchanges* [19].

Transactional exchanges involve an exchange of a set of resources rights. A typical example would be a sale exchange where the company relinquishes its *ownership right* on the product it is selling and gaining an *ownership right* on customer's money. Other types of rights may include usage rights, copyrights, etc. Thus, resources involved in transactional exchanges are perceived as a collection of rights. Compensating a transactional exchange requires us to reverse the exchange by returning those rights to their original provider. For example, compensating the exchange E4 from Fig. 2 would involve returning the ownership of the money to the customer and the ownership rights of the book to Amazon.

However, business activity do not revolve solely on transactional aspects. Although most business collaboration processes comprise a transactional activity at some point, they are usually combined with transformational activities that either *use/alter* or *consume* some resources in order to gain new or enhanced resources having an added value praised by the customer. These are called conversional exchanges and resources involved are defined by a set of properties (as opposed to a set of rights) being altered by the economic events.

Consumed resources cease to exist and cannot be restored to the provider agent. Thus, the provider agent will be compensated by receiving a resource of an equivalent value. We propose to compensate a consumed resource by relinquishing an abstract economic resource called *a claim* that could be settled in the future by an equivalent value - usually money. Claims have been introduced by McCarthy as “(...) not tangible resources for and against the enterprise. Claims derive from imbalances in duality relationships where an enterprise has either (1) gained control of a resource and is now accountable for a future decrement (...) or (2) relinquished control of a resource and is now entitled to a future increment (...)”[20]. In our example, the box has been consumed and is therefore compensated by a claim.

On the other hand, a used resource is one whose properties- or a subset thereof- are altered by the conversational exchange. Compensation consists then of attempting to restore those properties to their original values. For example, an REA event that constructs a machine by assembling its parts can be compensated by disassembling those parts. However, if some parts are altered during the assembly (e.g. welded), they may not recover their original value upon disassembly. Here again, the original provider of the part would need to be issued a claim for the lost value.

What is the Type of the Resource? I walk into a food store, pick-up a can of soup that has an expiration/consume-before date. If I change my mind and decide to return the can, the store will likely take it back if it is before the consume-before date. If I return it *after* that date, the can would have lost all of its value. The soup can belongs to the category of *perishable resources* whose value goes to zero at a given date/instant. Other perishable resources include hotel room bookings, flight seats, or rock concert tickets. If, on the other hand, the purchaser decides to return the resource *before* the consume-before date, the seller may credit them for *part of the purchase*, depending on how difficult it is for the seller to turn around and resell it while it still has value. This simple example illustrates one *dimension/subcategorization* of resource types that influences, a) whether or not a resource can be compensated, and b) how and how much. We identified four such (non-orthogonal) dimensions, explained below: *reversibility, perishability, depreciation, and discreteness*.

- Reversibility: Shipping the book from *Amazon's* warehouse to customer's location does not normally affect the book's physical condition. This means that the book will not sustain any value loss on its way to the customer. Consequently, we consider the book as being a reversible resource that we compensate by restoring its original property. However, the reversibility factor must be linked to the resource's property being altered by the event. Thus if, say, the same book has been autographed before being shipped (i.e. a conversion process altering its *autographed* property), then it has been irreversibly altered by the event. A non-reversible resource involves a loss in value, in full or in part, sustained by the owner, that a compensation process must take into account, for example by redeeming the lost value.
- Perishability and depreciation/appreciation: As explained above, resource is *perishable* if it completely loses its value at a given date. By contrast, *depreciation* corresponds to a *gradual* loss of value, notwithstanding the loss of value due to wear and tear. For example, notwithstanding generous product return policies, over time, computers and cars depreciate, even when they are not used, because of the advances in technology (and fashion). Conversely, art and collector items tend to increase in value, with time. Both perishability and depreciation/appreciation involve compensating for the resource's lost value as a function of time (i.e. time of process interruption).
- Discreteness: I need a four-foot long wooden beam, with a two-by-four inch base. The local hardware store sells only beams that are eight foot long. If I

cut and use a four-foot segment, I cannot return the other half. Similarly, if I need five four-foot segments, I will need to buy three eight-foot beams. With discrete resources, exchanges and compensation are measured in discrete units, even if the actual quantity used is continuous. By contrast, if I consume 5MW of power, I will pay for only— and exactly— that. Non-discrete resources tend to be substance-like [7], in the sense that if we divide it into two (or more) parts, the parts are of the same nature as the original resource.

Going back to our Amazon, the book is a) reversible on both its packaged and location properties, b) non perishable and c) discrete. Therefore, returning the book would not involve any loss in value to consider for compensation.

Is it a Gradual Event? Assume that I want to paint my living room a light sunny blue and I hire a painter to work by the hour. The painter needs to mix the paint first, to obtain a gallon-plus of light blue paint, and then paint the living room with it. The ‘mix the paint’ activity/REA event *consumes* the ‘input’ cans at the beginning, i.e. as soon as I pour *one drop* of color paint into the white paint container. Indeed, from that point on, neither can of paint can be reused nor repurposed. By contrast, the actual painting of the living room consumes the labor *gradually*: if the painter stops at any time during the activity, I will only pay for time used.

This example illustrates the difference between an activity (REA event) that consumes its input resources *gradually* (laying the paint on the walls) from one that consumes its resources *atomically* at some point during the activity, typically at the beginning (mixing the paint), or at the end.

Going back to our amazon example, for practical and business reasons, all of the activities (REA events) consume their resources atomically. For example, once we start packing the book (exchange E1 in Fig.2), the box is altered, and we consider that the labor required for the packing has been consumed. Similarly, once the book leaves the warehouse, we consider that the shipment service has been consumed.

Are there Event Costs? Many business activities, and REA exchanges in particular, involve labor. Theoretically, a value chain model should show labor as an economic resource, and represent its consumption by separate economic events. However, such a granularity of representation may result in large and hard to understand models. As discussed earlier in section 3.1, Geerts et al. argued that analysts should stop modeling at the level enabling to plan, control and evaluate the business process. It is not clear that accounting for labor consumption, *in all circumstances*, with separate economic events helps to plan, control and evaluate the business process. For example, in a car manufacturing process (a *conventional exchange*), labor accounts for a significant part of the cost, and it should be represented as a *resource*, and *its consumption* on the assembly line should be represented by an *economic event*. By contrast, approving supplier invoices for payment should not. This is *not* to say that these costs should not be counted. We suggest to use a ‘cost’ attribute attached to events, that aggregates

all of the costs that we deemed too low-level to merit a full REA treatment. If a completed activity is to be compensated, in addition to the resources that it did consume (reversibly or not, atomically or not, etc.), we need also to compensate for the activity cost.

In our example, the labor used for packing (exchange E1 in Fig. 2) is included in the 'cost' property of the exchange E1. Similarly, the banking transaction charges (exchange E4 in Fig. 2) are included in the 'cost' property of exchange E4.

Are there Compensation-Specific Business Policies and Rules? Going back to the process presented in Fig. 2, suppose that the customer decides to cancel his/her order after the package was sent out for shipping. The previous factors determine whether the book is returnable, and how much resources have been irreversibly used or consumed, both in the going forward process, and in the compensation process. There remain a number of issues / choices, which are typically driven by organization/company-specific policies or rules. Most notably:

1. should the resource(s) that is/are at the heart of the process (book and cash) be recoverable, in what form will they be returned to their original owners? in our amazon example, the customer may either have his credit card credited for the refund, or receive a credit voucher with the equivalent value towards future purchases
2. how much should the customer be credited, anyway? as we saw from our amazon example, the cancellation of a book order carries a number of non-reversible costs, including the box used for packing, the shipping to and back from the customer, the banking transaction costs, and the labor involved in packing and unpacking. It is a matter of business policy to choose which costs to incur to the customer, and under what circumstances (see 'who is the accountable agent' below).

We see business policies as business domain and corporate specific refinements of the compensation factors discussed above. A business policy does *not* change the *nature* of an event or the *type* of the resource. However, it may influence the choice of which compensation activity to choose, among many, or how much to compensate for.

In the amazon example, customers who cancel orders that have been shipped are liable for, a) the initial shipping charges, b) the return shipping charges. However, they are not liable for the labor costs or the banking transaction costs. Further, the money they are entitled to is credited back to their credit cards.

Instance Factors. The *class factors* introduced above enabled to determine the compensation activities required for a given exchange with given dual events and involving given resources. They are general factors applicable to any execution of the business process. Conversely, instance factors apply to a specific execution of the process and enable us to assess to which extent one should compensate for a given resource. We identify two instance factors: the time of interruption and the accountable agent.

What is the Time of Interruption? Abortion of a business process may happen anywhere between the instant it starts and the instant it ends. Thus knowing the time at which a process aborts is a critical information in order to establish (1) which of the activities completed and thus need to be compensated and (2) how much of the resources involved during the last running event(s) need to be accounted for.

Who is the Accountable Agent? If one cancels his trip a week prior to the departure date he expects to pay for cancellation fees that may go as far as the price of purchase for the trip. Indeed, aborting a business process may involve losses sustained by either the company in charge of the business process and/or an actor involved in the process. Some of these losses may be absorbed by the organization fully or partially as dictated by business policies while some won't, thus rendering one of the agents responsible for the losses. In order to identify the accountable agent of a given abortion, we conjecture him being:

- The provider agent of the economic event causing the process abortion in a case of process failure (e.g. The travel agency if the plane seats have been overbooked); or
- The agent who triggered the canceling event (e.g. the customer who cancelled his order).

3.3 Modeling Compensation Processes

The compensation factors presented in the previous subsection along with a given REA business process value chain will enable us to infer corresponding compensation activities for each REA exchange. Our methodology relies on a catalog of <REA exchange, REA compensation exchange> patterns based on the factors identified above that we will build. For example, we might have different pairs for the different types of conversions, based on the resource types, the gradualness of the consumption, etc. Note that the pure transactional exchanges are reversible, modulo the transaction costs (and if different, the reverse transaction costs). For conversion processes, the factors discussed above should help us design the REA compensation exchanges at a high level.

Given a value chain model, we consider every exchange and classify it according to the catalog outlined above by asking questions to the analyst. Such question may include: *Is your resource <A> perishable (discrete, etc.)? Does your event <E> consume its resources at the beginning, gradually or at the end?*. Based on the answers, we will be able to select the appropriate pattern/pair <REA exchange, REA compensation exchange>, and instantiate the REA compensation exchange part using her domain terminology. An illustration of such compensation exchange instantiation applied to exchange E1 from our *Amazon* example is shown in Fig. 3.

Finally, once the relevant compensation exchanges have been identified, we propose to construct a compensation value chain by composing compensation exchanges in reverse order following the Sagas[10] approach and in respect to the resource flow of the original value chain.

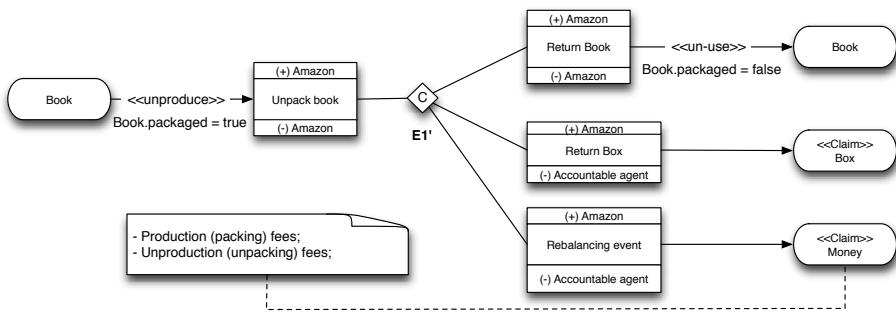


Fig. 3. Compensation exchange of E1 from Fig. 2

4 Discussion

4.1 Related Work

Our work builds on transaction management in the fields of distributed databases and long running workflows. Most of current implementations of error recovery in business process enactment engines rely on the Sagas[10] approach first introduced by Garcia-Molina. A saga is a chained transactions technique aiming at ensuring global transaction atomicity by slicing a process into a set of ACID transactions. In case of an error and the process needs to abort, successful ACID transactions are compensated in their reverse execution order by invoking their compensation handlers. WS-BPEL[22], a standard language for service composition execution, uses a Saga-like approach to handle errors by implementing *fault*, *compensation* and *termination* (FCT) handlers. In order to achieve model checking and ensure reachability in error handlers, authors (e.g. [9], [4]) expressed FCT mechanisms in formal semantics based on pi-calculus and Petrinets.

In recent work, authors proposed alternate approaches to handle process execution failures, mainly to achieve process self-healiblity. Techniques inspired by aspect programming(e.g. [14,26]) permitted to separate process design from failure handling by treating error paths as crosscutting concerns. Advances in semantic web services allowed implementing transactional support through negotiating agents([3]). Some approaches emphasize the human involvement in recovering from business process exceptions. In order to achieve organizational resilience, Antunes proposes a framework integrating both machine and human involvement in error recovery[1]. His approach relies on a *control switch* concept supporting ad-hoc human interventions by moving control out of the BPM enactment engine whenever a certain type of exception occurs. Although we find these directions promising, we argue that the problem of compensation remains to handle backward recovery. All the approaches above mentioned focus essentially on technical aspects from an operational perspective such as language constructs, message exchanges and coordination. To the best of our knowledge, no work has been done in supporting compensation design at the analysis level.

On the business process design, different authors stressed the advantages of business modeling prior to process modeling in order to assess the rationale of the business process and to express business objectives at a high level, from which implementation should be derived. Many business ontologies have been proposed that fall into two groups, value-oriented (e.g. [20,15,24]) and goal-oriented approaches (e.g. [27,2,17]). Although authors have tackled the problem of process modeling based on abstract business objectives, none has applied these aspects to error recovery and compensation. Thus our work aims at filling this gap and offers to consider business modeling as a valuable option to the error recovery.

4.2 Validation Approach

Generally speaking, there are two aspects of our approach we need to validate. Firstly, as mentioned in section 3.3, we will construct an exhaustive catalog of *<REA exchange, REA compensation pattern>* pairs based on our compensation factors. The catalog will be validated using existing business ontologies and on business process modeling experts who will be asked to assess the soundness of our high level compensation patterns.

The second aspect of our validation will concern the resulting compensation processes obtained using our methodology for which we will take a two steps approach. First, we will compare our results to a catalog of classified business processes. We chose the reference model of the SAP business blueprints[6] because we believe it is at the right level of abstraction enabling us to validate on a wide range of business domain activities. The reference model describes best-business practices from many different industries by modeling, both the 'happy' and alternate paths, of generic business processes. We will extract those happy paths and model their value chain using the REA ontology. Then we will walk through our methodology in order to generate compensation processes and compare them to the business error handling paths of SAP's model. We expect to generate more compensation processes than can be found in SAP blueprint because our method will cover the cases exhaustively. As a second step, we will validate on real world cases by consulting a panel of subject matter experts within specific organizations. Our approach will be to work with their business process and see whether (1) we ask the right questions (perishability, discreteness, etc.), and (2) the resulting compensation processes are sound from their business perspective.

Note that we have developed a prototype implementing our approach and applied our factors and our framework to a number of real world examples. This enabled us to refine our initial set of compensation factors and to establish the ground rules of our approach.

5 Conclusion

In this paper, we proposed a business-oriented approach in order to assist the business process designer in establishing the compensation activities. We argue that

despite apparent numerous ways of compensating for a given business process, the compensation is mainly a business problem; therefore, the solution should be tackled from a business standpoint.

Extensive research focused on the compensation from a technical programming perspective and, to the best of our knowledge, none has addressed the problem from a business standpoint. Our work aims at filling that gap. The main contribution of our work lies in establishing the decision factors involved in designing compensation activities. Relying on these factors, we were able to determine compensation activities and elicit their requirements in a systematic fashion, hence providing the business analysts with a much-welcomed guidance.

Although this work is still at an early stage, this paper focuses on establishing our ground ideas and our major directions. We are currently working on constructing a catalog of high level business activities paired with corresponding compensations in order to apply the compensation factors programmatically.

References

1. Antunes, P.: BPM and Exception Handling: Focus on Organizational Resilience. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41(3), 383–392 (2011)
2. Behnam, S.A., Amyot, D., Mussbacher, G.: Towards a Pattern-Based Framework for Goal-Driven Business Process Modeling. In: 2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications (SERA), pp. 137–145 (2010)
3. Bocchi, L., Ciancarini, P., Rossi, D.: Transactional aspects in semantic based discovery of services. In: Jacquet, J.-M., Picco, G.P. (eds.) COORDINATION 2005. LNCS, vol. 3454, pp. 283–297. Springer, Heidelberg (2005)
4. Butler, M., Ferreira, C.: A Process Compensation Language. In: Grieskamp, W., Santen, T., Stoddart, B. (eds.) IFM 2000. LNCS, vol. 1945, pp. 61–76. Springer, Heidelberg (2000)
5. Butler, M., Ferreira, C.: An Operational Semantics for StAC, a Language for Modelling Long-running Business Transactions. In: De Nicola, R., Ferrari, G.-L., Meredith, G. (eds.) COORDINATION 2004. LNCS, vol. 2949, pp. 87–104. Springer, Heidelberg (2004)
6. Curran, T., Keller, G., Ladd, A.: SAP R/3 Business Blueprint. Understanding the business process reference model. Prentice Hall (1998)
7. CYCorp. CYC Knowledge Base
8. Dayal, U., Hsu, M., Ladin, R.: Business Process Coordination: State of the Art, Trends, and Open Issues. In: 27th Very Large Databases Conference, VLDB 2001, Roma, pp. 3–13 (2001)
9. Eisentraut, C., Spieler, D.: Fault, Compensation and Termination in WS-BPEL 2.0 - A Comparative Analysis. In: Bruni, R., Wolf, K. (eds.) WS-FM 2008. LNCS, vol. 5387, pp. 107–126. Springer, Heidelberg (2009)
10. Garcia-Molina, H., Salem, K.: Sagas. In: SIGMOD 1987: Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, pp. 249–259. ACM Request Permissions, New York (1987)
11. Geerts, G.L., McCarthy, W.E.: Modeling Business Enterprises as Value-Added Process Hierarchies with Resource-Event-Agent Object Templates. In: Business Object Design and Implementation, pp. 94–113. Springer (1997)

12. Geerts, G.L., McCarthy, W.E.: The Ontological Foundation of REA Enterprise Information Systems. In: American Accounting Association Conference, Philadelphia (2000)
13. Geerts, G.L., McCarthy, W.E.: An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. International Journal of Accounting Information Systems 3(1) (2002)
14. Ghidini, C., Francescomarino, C.D., Rospocher, M., Tonella, P., Serafini, L.: Semantics-Based Aspect-Oriented Management of Exceptional Flows in Business Processes. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42(1), 25–37 (2012)
15. Gordijn, J., Akkermans, H., van Vliet, H.: Business Modelling is not Process Modelling. In: Mayr, H.C., Liddle, S.W., Thalheim, B. (eds.) ER Workshops 2000. LNCS, vol. 1921, pp. 40–51. Springer, Heidelberg (2000)
16. Gordijn, J., Wieringa, R.: A value-oriented approach to e-business process design. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 390–403. Springer, Heidelberg (2003)
17. Gordijn, J., Yu, E., van der Raadt, B.: E-service design using i* and e^{/sup} 3/ value modeling. IEEE Software 23(3), 26–33 (2006)
18. Greenfield, P., Fekete, A., Jang, J., Kuo, D.: Compensation is not enough (fault-handling and compensation mechanism). In: Proceedings of the Seventh IEEE International Enterprise Distributed Object Computing Conference 2003, pp. 232–239. IEEE Comput. Soc. (2006)
19. Hruby, P.: Model-Driven Design Using Business Patterns. Springer (2006)
20. McCarthy, W.E.: The REA Accounting Model - A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review 57(3), 554–578 (1982)
21. Mili, H., Godin, R., Tremblay, G., Dorfeuille, W.: Towards a Methodology for Designing Compensation Processes in Long-Running Business Transactions. In: Montreal Conference on eTechnologies, MCETECH 2006, Montreal, pp. 137–148 (2006)
22. OASIS. Web Services Business Process Execution Language (BPEL). OASIS (2007)
23. OMG. Business Process Model and Notation (BPMN). OMG (January 2011)
24. Osterwalder, A.: The Business Model Ontology. PhD thesis, Université de Lausanne École des Hautes Études Commerciales (2004)
25. Porter, M.E.: Competitive advantage: Creating and Sustaining Superior Performance, New York, USA (1985)
26. Sonntag, M., Karastoyanova, D.: Compensation of Adapted Service Orchestration Logic in BPEL 'n' Aspects. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 413–428. Springer, Heidelberg (2011)
27. Yu, E.S.K.: Models for supporting the redesign of organizational work. In: COCS 1995: Proceedings of Conference on Organizational Computing Systems. ACM Request Permissions (August 1995)
28. Yu, S.C.: The Structure of Accounting Theory. The University Press of Florida (1976)

An Effort Prediction Model Based on BPM Measures for Process Automation

Banu Aysolmaz¹, Deniz İren², and Onur Demirörs¹

¹ Informatics Institute, Middle East Technical University, Ankara, Turkey

{aysolmaz, demirors}@metu.edu.tr

² Computer Center, Middle East Technical University, Ankara, Turkey

diren@metu.edu.tr

Abstract. BPM software automation projects require different approaches for effort estimation for they are developed based on business process models rather than traditional requirements analysis outputs. In this empirical research we examine the effect of various measures for BPMN compliant business process models on the effort spent to automate those models. Although different measures are suggested in the literature, only a few studies exist that relate these measures to effort estimation. We propose that different perspectives of business process models need to be considered such as behavioral, organizational, functional and informational to determine the automation effort effectively. The proposed measures include number of activities, number of participating roles, number of outputs from the process and control flow complexity. We examine the effect of these measures on the automation effort and propose a prediction model developed by multiple linear regression analysis. The data were collected from a large IS integration project which cost 300 person-months along a three-year time frame. The results indicate that some of the measures collected have significant effect on the effort spent to develop the BPM automation software. We envision that prediction models developed by using the suggested approach will be useful to make accurate estimates of project effort for BPM intensive software development projects.

Keywords: Business process model measures, business process automation, project management, effort prediction model.

1 Introduction

Various measures have been suggested and utilized for business process models in the literature for different purposes such as; evaluating quality [10, 21], understanding the error probability [16], assessing the understandability and maintainability [8, 9, 20], measuring the similarity between models [7] and measuring functional size for the software to be developed based on the process models [12].

The business process models are commonly used as a tool to analyze requirements in early stages [2, 18]. When available, business process models contain valuable information on the size of the system to be automated in early phases of the development life cycle [12].

Only a few studies exist in the literature analyzing the size of business process models to be used as a basis of system development effort prediction [12, 18]. These studies suggest a methodology to determine the functional size of the software to be developed indirectly, by estimating the size with respect to well-known functional size measurement methods.

Currently, we develop an Integrated Campus Information System in Middle East Technical University (METU) based on the university's business processes. This project aims to automate most of the processes of the university including more than 90 process modules. For each process, business process analyses are being conducted and business processes are being defined as models and structured process definitions are being written. The project is divided into phases in which a predefined set of process modules are automated. For each of the phases, the project management office requires an estimation of the automation effort to use for planning, budgeting and subcontracting the project.

In this study, we explain our empirical research focusing on developing an effort prediction model which is based on direct measures on business process models. We aimed at identifying the effect of various business process model attributes to the automation effort and suggest an effort estimation model by using multiple linear regression analysis technique. The measure set we used covers different perspectives of business process models which are functional, behavioral, organizational and informational. We used the historical data gathered over the last three years covering approximately 300 person-months of effort and 10 business process modules of varying sizes which have been analyzed, modeled and automated into working software.

The remainder of the paper is organized as follows. In the following section, we provide a brief overview of the existing literature. In the following section, we describe the case, introduce the research problem together with the proposed solution, explain the case study plan and identify the selected set of business process model measures together with the rationale for selection. In the fourth section, we describe the empirical approach used including the data collected, the method applied and the results. The last section provides a discussion of conclusions, limitations and future work.

2 Related Work

There is an extensive body of literature on definition and discussion of various business process model measures. Business process measures are usually derived from software metrics [1, 10, 21, 30]. Process size is often determined by using a corresponding software size metric; "Line of Code". Some of the size measures used are; number of activities, joins and splits [1, 16], diameter, density [16], and cross-connectivity [22].

Another basic measure derived from software measurement domain is complexity. Cyclomatic Complexity is usually translated to business process models as Control Flow Complexity (CFC) where different types of splits are handled separately [21].

Other complexity measures have also been suggested such as; Halstead-based Process Complexity, Coefficient of Network Complexity [17], nesting depth, jump-outs from control structure, cognitive complexity [9]. We observed that CFC is the most commonly used complexity measure for business process models [1, 9, 21]. It is important to mention that CFC only covers control-flow complexity, and disregards complexity that might have been introduced because of data elements captured within process models.

Fan-in and fan-out metrics are used in different ways like inputs and outputs [17] and reference numbers to and from a process module [9]. Measures like coefficient of connectivity, separability, sequentiality, depth, structuredness were suggested to measure error proneness of the process models [16, 30].

There are other measures that attempt to involve data complexity. Interface complexity measure [1] includes number of inputs and outputs in the process. Coupling refers to the number of activity couples which contain one or more common data elements and cohesion expresses the coherence within the activities of the process model [17]. Dhammaraksa and Intakosum [6] emphasize that current size measures do not consider all perspectives of process models; namely functional, behavioral, informational and organizational. In their research they provide measures for each perspective.

Many studies include further analysis and empirical work to use measures to evaluate error probability in process models, understandability and complexity. Mendling et al. [16] show that there is a strong connection between formal process model errors and a set of measures on structural and behavioral aspects of process models. In another research they emphasize that there is a negative correlation between size and quality aspects. Mendling et.al. [23] illustrate that higher density of arcs and larger number of paths in a model affects the understandability negatively.

We observe that many measures are suggested to measure the size of business process models. But these measures are collected for other purposes, like analyzing understandability and error-proneness of models. There are a few research published just in recent years focusing on business process size in order to evaluate its effect on development effort of related software systems by using COSMIC model.

COSMIC is a widely accepted method for functional size measurement of software, and accepted as an international standard [3, 11]. An extensive survey on conceptual model based functional size measurement research [15] revealed that all studies calculate size from UML diagrams. We found other studies focusing on using business process models to measure COSMIC size. Lavazza and Bianco [14] used UML activity diagrams to estimate the COSMIC functional size of the system. Their approach is indirect effort estimation method from business process models.

Monsal and Abran [18] defined a set of BPM rules for software to be developed, the users of the software and data movements. By conforming to these rules during modeling, it is assured that COSMIC size measurement can be calculated. The authors developed rules for Qualigram notation (their own modeling notation) and BPMN. Another important study by Kaya and Demirörs [12] describes the size measurement method based on EPC notation. For each function in EPC, further analysis is conducted (namely FAD) to determine data movements, users and related systems.

With this method, COSMIC size is calculated by counting data movements for related systems. The approach is also automated on a modeling tool. These approaches aim to measure the size of the system to be automated based on process models, rather than determining measures of business process size that can be used to identify the automation effort.

3 The Case Study

3.1 Description of the Case

METU has 24.000 students currently enrolled. There are 40 undergraduate departments and 160 graduate programs. METU also has 21 interdisciplinary research centers. More than 5.000 personnel are working for the university. A large number of IT systems have been developed since the establishment of the university Computer Center. However most of these IT systems run independently, not communicating with each other and using various technologies for data storage and communication. As a result of this crowded, complex environment, problems emerged such as out-of-control duplication of data, non-standard communication, lack of control over IT service levels and very high maintenance costs.

The Integrated Campus Information System (ICIS) project was initiated in 2009 by Computer Center in order to solve these problems. ICIS aims at integrating the existing IS applications in accordance with the university strategic plan. Initially Computer Center developed a business process map consisting of all the business processes of METU. These processes were prioritized in line with the master project plan.

Since the beginning of the project, the activities of analyzing and modeling of business processes and eventually developing software running on automated business process models are conducted iteratively for each process module. More than 300 person-months are utilized in the last 3 years.

The method for developing process automation can be summarized as follows. First all stakeholders involved in the business process are contacted. Then modeling experts start analysis sessions with the stakeholders. The analysis team puts in additional effort to produce process definitions documentation. The process definition documents contain textual definitions of processes organized in sets and subsets, stakeholders, business rules, risks, inputs and outputs of the process and data elements. Then the BPMN models of the processes are developed in accordance with the textual definitions. The data elements are fed into the university data dictionary. In compliance with the data element definitions, the web services are implemented and SOA mediation layer are integrated with them. Then the software models associated with the process model are developed. After functional testing the process automation software to the university portal are integrated via the user interface portlets.

The models are developed by using Eclipse BPMN modeler, in compliance with BPMN 2.0 standard. Activiti is utilized as the underlying process engine. Business rules are represented in Drools. All programs are coded with Java, JSP and Javascript.

Considering this software lifecycle, we observe that work outputs produced in BPM software automation projects differ from traditional software artifacts. Usually, in addition to traditional artifacts like software requirements specification and software design description documents, business process models are developed. These models are utilized as the basis for the development activities, as the code is produced focused on each activity presented in the process model.

There are about 90 process modules identified in the business process map. The concept of “process module” is used for a group of process models which are coherent and focus on a specific working area of the organization (like budgeting). All of the process models under a process module are connected to the hierarchical structure of the related module. Until now, the team completed developing 10 modules and working on 15 modules at the moment. We were able to collect both business process model measures and total development effort for the 10 modules completed. The university needs to make effort estimation for the rest of the modules to be developed in phases to use as the basis of planning and budgeting.

3.2 Problem Statement and Proposed Solution

Effort estimation for software development is a critical activity for project planning. Widely accepted and commonly used effort estimation methods exist for software development activities producing traditional requirements analysis outputs, like COSMIC function point measurement. BPM software automation projects require different approaches for effort estimation for they are developed based on business process models. Early outputs of the development lifecycle for this kind of projects are business process models which embed information on the requirements of the system. There are a few studies that utilize business process models to determine COSMIC size measurement of the system to be automated. However, to our knowledge, the effect of different business process model properties on the development effort is not studied.

The project management office in METU Computer Center requires all projects to have an effort estimation method based on something more than educated guess that utilizes their existing experience from the project. ICIS project is the first and the only project carried out in the university’s Computer Center in which business process automation is used. Our experience supports our previous observations that the work products of process automation projects differ from traditional software development projects. So, in order to be able to estimate the project effort for upcoming phases of the ICIS project (and potentially other process automation projects), the need for a viable and empirical estimation method emerged.

We were not able to derive an extensive conclusion from the literature for what possible business process model properties we can use to build up an effort estimation method for our case. Thus, the first step of our proposed solution is identification of a set of business process model measures that may have effect on the automation effort. Then, we propose to perform statistical analysis on the collected data to identify the measures that have significant impact on the effort; determine the effect of each measure and formulate an effort prediction model.

3.3 Case Study Plan

The basic procedure common for effort prediction is applied in this study. Following tasks are envisaged within the case study plan:

- Identify the measurable properties of business process models and select the set of measures to be collected that are anticipated to have effect on automation effort,
- Collect data on the selected case; e.g. collect the data for selected measure set and measure the effort spent to automate the processes for each process module,
- Correlate the data collected on business process model measures to the related effort measurements and develop an effort prediction model by multiple linear regression analysis,
- Analyze the effect of each measure on the automation effort and update the model,
- Evaluate the prediction model considering the statistical results and by comparison of the predicted values to real values.

Apparently, many attributes of business process models will have effect on automation effort. But in this study, we specifically aim at identifying the size of the effect in addition to the measures with strongest effect on the development effort. For this, we identify process model measures that compose a sound set to reflect basic size properties of business process models. The selected measures and rationale for selection are explained in the below section.

3.4 Selection of the Business Process Model Measure Set

There are various measures for business process models that may have influence on the automation effort. Our aim in this empirical study is to reveal a set of business process measures that can largely explain the automation effort and suggest an effort prediction model by means of linear regression analysis. These measures shall not be highly correlated to each other and shall represent the business process models meaningfully in BPM domain.

As Curtis et.al. state [4] that to adequately describe a process, four different perspectives shall be taken into consideration: functional, behavioral, organizational and informational. In this research, we consider that our measure set shall cover information regarding each perspective so that we can find out specific contribution of each perspective on the automation effort.

The data is collected from a set of business process models that are developed by the same group of process modeling experts consistently using the same modeling methods. Considering this, we also assume that all of the models conform to basic modeling principles (like block structure [13]), and there is not much deviation between error-proneness and style of the models. Moreover, our aim is not to evaluate how “good” the models are. The nesting depth, which is found to be an important measure for complexity [9], is also similar among models. There are no models with more than 2 levels of nesting depth. Thus we scope out the quality and model structure measures like structuredness, separability, sequentiality [16, 13, 30], correctness of models and nesting depth.

We neither included an extra measure for data complexity like interface complexity suggested by Cardoso et.al. [1] nor other complexity measures such as Halstead-based Process Complexity, Coefficient of Network Complexity measures. We avoided utilizing derived measures and we identified base measures covering each model perspective so that we can observe their effect individually as a result of our linear regression analysis. If we had used derived measures, the high collinearity between independent variables of the regression would have reduced the reliability of the results.

As the initial set of measures, for functional perspective, we selected “number of activities”. We also considered using “number of human tasks” and “number of automated tasks” embedded in our BPMN notation. But to prevent dependency on modeling notation we decided on only choosing number of activities.

We also considered utilizing fan-out measure as an indicator of number of subprocesses referenced from a process diagram. This measure can also add value for representing the functional perspective. However, due to the nature of processes modeled so far in this project there were only a few references to other processes. Thus we were not able to use fan-out as a potential predictor of effort in this research.

Behavioral perspective of a process model represents sequencing and possible alternative flows of the model. Cyclomatic complexity (CC) which is the basic complexity measure for software engineering domain covers complexity introduced by decision points. Control Flow Complexity (CFC) applies a different calculation for each decision point type (and, or, xor). We calculated both measures for our model, but as a result of our statistical analysis we observed that they are highly correlated to each other and CFC has more power on explaining the effort variable. Considering that, we dropped CC and just used CFC for further analysis.

Informational perspective represents the informational entities “produced or manipulated” by the process. To present this perspective, we selected to analyze three measures: number of outputs, number of inputs and total number of inputs and outputs. Although the inputs are used by functions of the system, whole data covered by inputs may not be representing the manipulated entities. As our initial statistical analysis also supported that the number of outputs are better correlated with the effort, we dropped the other measures and used only the number of outputs.

Organizational perspective represents the performers of the activities in process models. To depict this perspective, we chose “the total number of roles” in each process model.

We considered using other measures that may affect the automation effort; like number of business rules, number of performance indicators. As our sample size is limited and these measures are not about the key aspects of models, we left the analysis of these measures for future work.

For reasons provided we selected the measures that are defined below. It should be noted that each process module is composed of multiple subdiagrams. To calculate the following measures, the values of individual subdiagrams are added up as explained below.

- **Number of Activities:** NOA refers to the total number of activities (human and automated) in all subdiagrams of the process module.
- **Control Flow Complexity:** CFC is the sum of all complexity introduced by each split in the models. Joins are not considered in the measure. For each XOR split, the number of splits are counted, and for each AND split, “1” is added to the count. (there are no OR element in the models). The CFC values of all subdiagrams are added up.
- **Number of outputs:** NOO refers to the total number of output elements produced within the process module.
- **Number of roles:** NOR is the number of all roles performing in the related process module.

To summarize, in measure selection, we aimed at identifying at least one measure for each process perspective; functional, behavioral, informational and organizational. These measures are chosen to represent one of these perspectives directly, with the least possible collinearity between each other. That is the reason we focused on identifying base measures rather than derived measures. Additionally, due to the nature of the processes and the aim of the study, we scoped out quality and model structure metrics.

The process models used in this research are developed using BPMN notation [19]. However, the set of symbols used and the measures are common to many business process modeling languages, therefore the results can be applied to other languages.

4 Case Study Implementation

In this section we explain the data collected in this research and the methodology to develop an effort prediction model by using the defined set of process model measures. We conduct linear multiple regression analysis in which we utilize process model measures as independent variables and automation effort as the dependent variable.

4.1 Data

There are 10 completed process modules in the ICIS project for which automation effort (in person-months) was collected. These modules, focusing on research and financial management processes of the university, are considered large, as the average effort for development is 13 person-months. We cannot decompose the effort data into smaller pieces of process modules, so we need to utilize the data from 10 process modules as our sample data. We collected four process model measures for each of these modules as determined in previous section. We are aware that the sample size should be larger to attain more generalizable results. However, due to the nature of software development projects, it is difficult to collect data for large sample sizes. Given that this data have been collected over a three-year period, even the sample size is low, the collected data is still precious and can be utilized to draw conclusions.

The independent variables (NOA, CFC, NOO and NOR) are measured for each process module by examining each process model diagram and process definition documents. The effort data was collected and monitored for each process module as part of project management activities. The effort data covers all activities in the project, including process analysis, modeling and automation. As we think that the activities of process analysis and modeling are part of requirements analysis and software design; we also included their effort in our measurements.

Before starting the regression analysis, the data is examined for outliers. Only one outlier was observed for NOO measure. The related process module is examined and it is concluded that it is a special case because of the nature of the module; and the value is normalized in order not to affect the results. The data collected for the ten sample, including independent variables NOO, CFC, NOA, NOR and the dependent variable effort can be seen in Table 1.

Table 1. Collected Data

NOO	CFC	NOA	NOR	Effort (person-months)
3	9	15	5	5
1	8	13	2	8
2	13	23	4	3
12	27	61	6	22
13	37	100	13	18
2	0	9	4	3
9	74	167	20	20
5	63	72	16	24
2	2	24	11	4
11	30	35	13	22

4.2 Application of Multiple Linear Regression Analysis Method

In our study, we want to examine the effect of the independent variables (observe the correlation of these variables between each other, if exists) and fit a predictive model to our data to use for prediction of our dependent variable; effort. We assume that there is a linear relation between each of the independent variables and the dependent variable. The rationale behind this assumption can be explained; as work products –in this case process models- grow in size and complexity, the automation effort increases in a linear way. We already know from our experiences and due to the nature of the process modeling and software development activities that all of our process measures are positively correlated with the effort. To further analyze the effect of each measure and to identify their impact on the automation effort with a prediction model, multiple linear regression analysis method was applied.

The process measures are expected to be positively correlated to the effort, and if all of the measures are zero (e.g. there is no process model at all), the effort will be zero, too. This is why we forced the regression coefficients to be calculated through the origin and we have intercept value of zero.

In the first step, all of the four measures are added to the regression analysis. We observed very high collinearity statistics (> 0.9) between CFC and NOA. This high collinearity also caused the regression results to be insignificant. Further analysis of correlation between CFC and NOA supported that NOA is highly dependent on CFC. We dropped NOA from our regression analysis, as we concluded that we already cover information provided by NOA measure by means of CFC.

Conducting further analysis with the three measures; CFC, NOA and NOR, we observed that the regression coefficient for NOR is insignificant and the value of the coefficient is also very low. We concluded that the effect of number of roles (NOR) measure is insignificant for our prediction model; and we cannot identify a meaningful coefficient with our limited sample data. As a result, we dropped this measure from our model.

With the two independent variables; CFC and NOO; we ran our regression model and observed statistically significant results. The results are provided in the following section.

4.3 Results

A linear multiple regression model was used to develop an effort prediction model from the process model measures. The potential predictor measures used in the analysis were; Control Flow Complexity (CFC) and Number of Outputs (NOO). Table-2 provides descriptive statistics, where effort is shown in person-months. Table-3 shows zero order correlations which are statistically significant ($p < .01$) along with regression coefficients which are also statistically significant ($p < .05$).

Our prediction model was able to account for a 79.8% of the variance in automation effort, $F(2,7)=18.806$, $p < .01$, Adjusted $R^2=.798\%$. The mean magnitude of relative error (MMRE) of the model was calculated as $MMRE=30.20\%$. The prediction quality of our model was $PRED(30)=0.60$. This means that by using the prediction equation with inputs of NOO and CFC values, the model can predict the effort for 60% of the sample with less than 30% deviation.

Table 2. Descriptive Statistics

	Mean	Std.Dev.	N
Effort	12,90	8,9	10
NOO	6,00	25,4	10
CFC	26,30	4,7	10

Table 3. Regression coefficients and zero order correlations

Model	Beta	Sig.	Correlations	
			Zero-order	Sig.
NOO	,528	,025	,806	,002
	,513	,022	,813	,002

a. Dependent Variable: Effort

5 Conclusions, Limitations and Future Work

5.1 Conclusions

In this empirical research, we proposed an approach to determine a business process model measure set that can affect the automation effort of the related process models. Then, we conducted a linear multiple regression analysis on a set of data collected from Integrated Campus Integration System (ICIS) Project, which is a process automation project running over 3 years with more than 300 person-months of effort. The independent variables are selected to be four measures representing different perspectives of BPM; Number of Activities (NOA), Control Flow Complexity (CFC), Number of Outputs (NOO) and Number of Roles (NOR); and the dependent variable is the automation effort.

The resulting prediction model was able to explain a large amount of variance on the effort which was caused by the predictor variables (79.8%). The analysis results were statistically significant with two independent variables; CFC and NOO having almost the same effect on the automation effort. Also the evaluation of predictive quality of our model was performed by using mean magnitude of relative error and PRED(q) criteria, which is a widely accepted way of determining estimation quality in software engineering domain [24]. According to Conte et al. the it is desirable to have PRED(25) calculation over 0.75. Our model fails to achieve this level of prediction quality, which we explain with the low level of precision due to the lack of enough samples.

Through the steps of statistical analysis, we have scoped out NOA and NOR measures from the analysis. This does not mean these measures have no effect on the automation effort. We observed high collinearity between NOA and CFC. Our early analysis results showed CFC had larger effect on the effort. The primary cause of this result is that, CFC not only expresses the size but also the complexity of the model; meaning that CFC measure already covers the effect of NOA measure, resulting in no need to use NOA as an additional measure. We also observed that the NOR measure had relatively smaller effect on the effort thus omitted. However we suggest that this effect might be observed with better accuracy given a larger sample.

As a result of our linear regression model, we conclude that CFC and NOO have the most significant effect on the automation effort among our BPM measures; and both have almost the same weight of effect.

5.2 Limitations and Future Work

Small sample size in statistical analysis is a major threat to the validity of this research [5]. However, it is a well-known fact that larger samples are sometimes very difficult and time-consuming to establish in software domain [25]. Usage of statistical methods even with small sample size is common in software engineering domain [26, 27, 28, 29]. The result of such cases should be dealt with caution.

The most important limitation of our research is the small sample size. Still, we pursued our studies considering that the measurements made over the three years of the project covering more than 300 person-months of effort.

During the initial phases of statistical analysis, we dropped the measures; number of inputs and total number of inputs and outputs from the model; as we observed better correlation with number of outputs (NOO) measure. Based on our previous studies [12], we foresee that we could achieve better correlation with informational perspective of BPM if we could identify the total number of data movements; which is planned as the future work.

We plan to continue gathering data the same way and form a larger historical data pool in this project. With a larger sample size, we foresee we can attain more significant results for the existing measures and we can further examine the effects of more specific measures such as number of human tasks, number of automated tasks, fan-in, fan-out and number of business rules on development effort.

The current studies which utilize business process models for effort estimation, focus on some specific properties of models; especially informational perspectives [12, 18]. One of these studies is conducted by our research group [12]. We plan to use the results of this empirical research to extend our previous study by inserting different properties of business process models in BPM size and automation effort estimation method. We also believe that the approach and results of this study will contribute to other researchers to consider how various BPM properties may affect the automation effort.

Usage of statistical research methods in software engineering and business process management domains is rare; especially when the case requires data on effort. This is mostly caused by the difficulties of collecting a large sized sample. In this research we found the opportunity to utilize statistical methods and reach meaningful results. Thus we see this research is important because it exemplifies the usage of statistics as a powerful research method in business process modeling domain. We will be using the prediction model developed in this study for effort estimation of following phases of the project. We plan to replicate the same analysis as the ICIS project progresses in upcoming years to evaluate our prediction model as well as our research method.

We think that this research is important as it utilizes empirical outputs to support the development of estimation models; and we expect enhancements in the model with the collection of more data. We foresee that researchers and practitioners can benefit from this approach by following an analysis similar to the one described in this paper to formulate their prediction model coefficients using their own historical data.

References

1. Cardoso, J., Mendling, J., Neumann, G., Reijers, H.A.: A Discourse on Complexity of Process Models. In: Eder, J., Dustdar, S. (eds.) BPM 2006 Workshops. LNCS, vol. 4103, pp. 117–128. Springer, Heidelberg (2006)
2. Coskuncay, A., Aysolmaz, B., Demirörs, O., Bilen, O., Dogan, I.: Bridging The Gap Between Business Process Modeling And Software Requirements Analysis: A Case Study. In: MCIS 2010 Proceedings, paper 20 (2010), <http://aisel.aisnet.org/mcis2010/20>
3. COSMIC: The COSMIC Functional Size Measurement Method Version 3.0.1, Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003). The Common Software Measurement International Consortium (COSMIC) (2009)
4. Curtis, B., Kellner, M., Over, J.: Process Modeling. Communications of the ACM, Special Issue on Analysis and Modeling in Software Development 35(9), 75–90 (1992), doi:10.1145/130994.130998
5. Myrtveit, I., Stensrud, E., Shepperd, M.: Reliability and Validity in Comparative Studies of Software Prediction Models. In: IEEE Transactions on Software Engineering 31(5), 380 (2005)
6. Dhammaraksa, K., Intakosum, S.: Measuring Size of Business Process From Use Case Descriptions. In: Computer Science and Information Technology, ICCSIT 2009, pp. 600–604. IEEE (2009)
7. Dijkman, R., Dumas, M., van Dongen, B., Kaarik, R., Mendling, J.: Similarity of Business Process Models: Metrics and Evaluation. Information Systems Journal 36, 498–516 (2011)
8. Ghani, A.A.A., Wei, K.T., Muketha, G.M., Wen, W.P.: Complexity Metrics for Measuring the Understandability and Maintainability of Business Process Models Using Goal-Question-Metric (GQM). IJCSNS International Journal of Computer Science and Network Security 8(5) (2008)
9. Gruhn, V., Laue, R.: Approaches for Business Process Model Complexity Metrics. In: Abramowicz, W., Mayr, H.C. (eds.) Technologies For Business Information Systems, pp. 13–24 (2007), doi:10.1007/1-4020-5634-6_2
10. Guceglioglu, A.S., Demirörs, O.: Using Software Quality Characteristics to Measure Business Process Quality. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 374–379. Springer, Heidelberg (2005)
11. ISO/IEC: 19761:2003 Software engineering- COSMIC-FFP- A functional size measurement method. International Organization for Standardization, Switzerland (2003)
12. Kaya, M., Demirörs, O.: E-Cosmic: A Business Process Model Based Functional Size Estimation Approach. In: 37th EUROMICRO Conference Software Engineering and Advanced Applications, SEAA, pp. 404–410 (2011)
13. Laue, R., Mendling, J.: Structuredness and its significance for correctness of process models. Information Systems and E-Business Management 8(3), 287–307 (2010), doi:10.1007/s10257-009-0120-x
14. Lavazza, L., Del Bianco, V.: A case study in COSMIC functional size measurement: The rice cooker revisited. In: Abran, A., Braungarten, R., Dumke, R.R., Cuadrado-Gallego, J.J., Brunekreef, J. (eds.) IWSM 2009. LNCS, vol. 5891, pp. 101–121. Springer, Heidelberg (2009)
15. Marín, B., Giachetti, G., Pastor, Ó.: Measurement of Functional Size in Conceptual Models: A Survey of Measurement Procedures Based on COSMIC. In: Dumke, R.R., Braungarten, R., Büren, G., Abran, A., Cuadrado-Gallego, J.J. (eds.) IWSM 2008. LNCS, vol. 5338, pp. 170–183. Springer, Heidelberg (2008)

16. Mendling, J.: Validation of Metrics as Error Predictors. In: Metrics for Process Models. LNBP, vol. 6, pp. 135–150. Springer, Heidelberg (2009)
17. Mendling, J.: Metrics for Business Process Models. In: Metrics for Process Models. LNBP, vol. 6, pp. 103–133. Springer, Heidelberg (2009)
18. Monsalve, C., Abran, A., April, A.: Measuring Software Functional Size from Business Process Models. *International Journal of Software Engineering and Knowledge Engineering* 21(3), 311–338 (2011), doi:10.1142/S0218194011005359
19. OMG, OMG business process model and notation (BPMN), version 1.2 (Object Management Group) (2009)
20. Reynoso, L., Rolón, E., Genero, M., García, F., Ruiz, F., Piattini, M.: Formal Definition of Measures for BPMN Models. In: Abran, A., Braungarten, R., Dumke, R.R., Cuadrado-Gallego, J.J., Brunekreef, J. (eds.) IWSM 2009. LNCS, vol. 5891, pp. 285–306. Springer, Heidelberg (2009)
21. Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H., van der Aalst, W.: Quality Metrics for Business Process Models. In: Fischer, L. (ed.) 2007 BPM & Workflow Handbook, Workflow Management Coalition, Lighthouse Point, Florida, USA, pp. 179–190 (2007)
22. Vanderfeesten, I., Reijers, H.A., Mendling, J., van der Aalst, W.M.P., Cardoso, J.: On a Quest for Good Process Models: The Cross-Connectivity Metric. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 480–494. Springer, Heidelberg (2008)
23. Mendling, J., Reijers, H., van der Aalst, W.: Seven Process Modeling Guidelines (7PMG). *Information and Software Technology Journal* 52, 127–136 (2010)
24. Conte, S., Dunsmore, H., Shen, V.: Software Engineering Metrics and Models. Benjamin-Cummings, Menlo Park (1986)
25. Frakes, W.B., Succi, G.: An industrial study of reuse, quality, and productivity. *Journal of Systems and Software* 57(2), 99–106 (2001), doi:10.1016/S0164-1212(00)00121-7, ISSN 0164-1212
26. Moseley, C.W.: A timescale estimating model for rule-based systems. Ph.D. diss., North Tex. State Univ. (1987)
27. Rainer, A., Hall, T.: Key success factors for implementing software process improvement: a maturity-based analysis. *Journal of Systems and Software* 62(2), 71–84 (2002), doi:10.1016/S0164-1212(01)00122-4, ISSN 0164-1212
28. Potok, T.E., Vouk, M., Rindos, A.: Productivity analysis of object-oriented software developed in a commercial environment. *Software – Practice and Experience* 29(10), 833–847 (1999)
29. McBride, T., Henderson-Sellers, B., Zowghi, D.: Software development as a design or a production project: An empirical study of project monitoring and control. *Journal of Enterprise Information Management* 20(1), 70–82 (2007)
30. Muketha, G.M., Ghani, A.A.A., Selamat, M.H., Atan, R.: A Survey of Business Process Complexity Metrics. *Information Technology Journal* 9(7), 1336–1344 (2010)

On the Use of Goal Models and Business Process Models for Elicitation of System Requirements

Jose Luis de la Vara¹, Juan Sánchez², and Oscar Pastor³

¹ Certus Centre for Software V&V, Simula Research Laboratory, Norway

² Dept. Sistemas Informáticos y Computación, Universitat Politècnica de València, Spain

³ Centro de Investigación ProS, Universitat Politècnica de València, Spain

jdelavara@simula.no, jsanchez@dsic.upv.es, opastor@pros.upv.es

Abstract. Goal modelling and business process modelling are two techniques that can be used for elicitation of system requirements of an information system. In general, goal-based approaches aim at supporting the objectives that an organization needs to achieve, whereas business process-based approaches aim at supporting the activity of an organization. Consequently, it could be assumed that these two types of approaches represent completely different perspectives for elicitation of system requirements. In this paper we argue that a correspondence exists between the perspectives and that they can be considered equivalent in some operational aspects. Therefore, the use of a perspective also implies support for the other. This argument is based on the definition of a set of guidelines that shows how a goal model can be derived from a business process model. As a result, we discuss when selection of one of the perspectives or their combination would be more suitable for requirements elicitation.

Keywords: goal modelling, business process modelling, requirements elicitation, requirements engineering, information system.

1 Introduction

Requirements elicitation is the first activity of the requirements engineering (RE) process. This activity aims at discovering the purpose of a software system, which is later refined and mapped into system requirements. When having to elicit the system requirements of an information system (IS) for an organization, different techniques and types of approaches can be used [25, 36]. For example, goal modelling and business process modelling can be used for elicitation of system requirements. They have also driven many research efforts and been applied in industry [17, 31, 45].

Goals have long been recognized to be essential components of the RE process [45]. They can be defined as objectives that a software system should fulfil in order to meet stakeholders' needs. Therefore, goal-based RE approaches for elicitation of system requirements mainly aim at developing ISs that support the objectives that an organization needs to achieve by modelling and analysing its goals. Examples of well known goal-based RE approaches are i* [47], KAOS [45], and Map [39].

A business process is a set of structured and ordered activities that are performed in an organization to achieve some business goal [10]. A business process takes inputs

from the business environment and creates outputs, and is executed coordinately and dynamically by people and/or technical components that exchange information. Therefore, business process-based RE approaches for elicitation of systems requirements mainly aim at developing ISs that support the activity of an organization by modelling and analysing its business processes. Examples of well-known business process-based RE approaches are EKD [5], ARIS [41], and some based on UML [13].

Both goal modelling and business process modelling deal with business requirements (aka early requirements) for elicitation of system requirements and can be very important for IS development. For example, business/IT alignment is reached when business goals, activities, and processes of an organization are in harmony with the technology that supports them [30]. However, it could be considered that goal-based RE approaches and business process-based ones are completely different and that no direct correspondence exists between them because of the explicit focus on different aspects of the application domain (objectives vs. activities). Indeed, existing research that has dealt with derivation of business process models from goal models (e.g., [26]) has had to extend goal models with business process-oriented details.

In this paper, we discuss the correspondence that exists between goal models and business process models for elicitation of system requirements. For this purpose, we present a set of guidelines that allow derivation of a goal model from a business process model without providing extra information. The guidelines are based on patterns that can be found in business process models.

As a result, we show how both models can be considered equivalent in some operational aspects, thus elicitation of system requirements from business process models also implies support for organizational goals, and vice versa. In addition, we discuss when combination of goal and business process models or use of one of these techniques would be more suitable. This is useful in practice when having to decide upon their use. To our knowledge, this issue has not been addressed in literature yet.

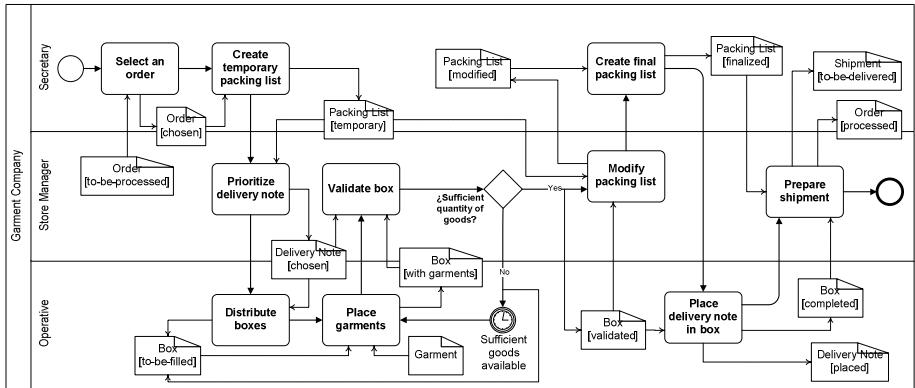
The rest of the paper is organized as follows. Section 2 reviews background work. Section 3 presents how a goal model can be derived from a business process model. Section 4 discusses their correspondence. Finally, Section 5 presents our conclusions.

2 Background

This section presents the background work on which the paper is based. First, operational goals in business processes are discussed. Next, related work is reviewed.

2.1 Operational Goals

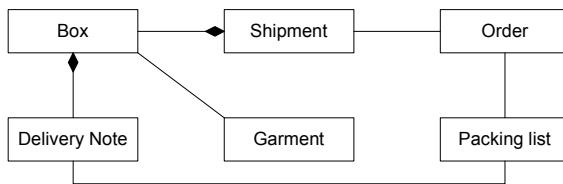
Business processes have goals that must be fulfilled during their execution [24]. There are sub-goals that denote milestones within a business process and whose fulfilment is possible due to the actions of all the participants involved [35]. These sub-goals are called operational goals, and indicate when an instance of a business process (model) can be considered completed [2]. Therefore, an operational goal is an objective or state that is or may be reached in a business process and indicates its completion.

**Fig. 1.** Example of BPD

In most of the notations for business process modelling (e.g., BPMN [33]), operational goals are implicitly declared in the structure of a business process model and the states of its resources and data entities. These entities and resources are input or output of the activities of a business process, and their states can change during the execution of the business process. As an example, Fig. 1 shows a BPD (Business Process Diagram, a business process model in BPMN). A description of the business process is not provided due to page limitations. It can be found in [10].

Since operational goals are implicitly part of a business process model, then a business process model can be considered equivalent to a goal model at least in some aspects. Therefore, a goal model can be derived from a business process model. Nonetheless, the correspondence between the models must be determined. If such a correspondence is found, then a business process model could be mapped into a goal model from patterns of the business process.

In addition to a business process model, a domain data model (Fig. 2) may be necessary for derivation of a goal model. This model includes (1) the entities that are used in a business process and whose states change as a result of its execution, and (2) the relations between the entities (associations and aggregations).

**Fig. 2.** Example of domain data model

2.2 Related Work

To our knowledge, this is the first paper that explicitly discusses the correspondence between goal models and business process models for elicitation of system requirements of ISs. Nonetheless, we are aware of works that have dealt with both types of models. These works are reviewed in this section.

In previous work, we dealt with elicitation of use cases from business process goals [11]. Although we also addressed derivation of a goal model from a business process model, the guidelines provided were not complete because some patterns of a business process model were not considered. A similar approach is presented in [7]. The authors proposed the concept of intentional fragments in BPDs as a set of elements of a process with a common purpose. These fragments are structured in the form of a KAOS model. However, the set of heuristics defined is limited if compared to, for instance, the number of guidelines presented in this paper.

In [12], we combined BPDs with Map models in order to represent the As-Is situation of an organization and analyse the strategic goals that an IS must help the organization to achieve. As a result, BPDs may change (To-Be situation). Task descriptions can then be elicited from BPDs.

Well-known business process-based RE approaches such as EKD [5], ARIS [41], and some based on UML [13] combine business process models and goal models by specifying the business goals that are fulfilled by executing a given business process. This is probably the most frequent way to combine goal models and business process models, and it can be found in other works such as [3, 19, 24, 37]. Guidance for discovering goals from scenarios and vice versa has also been proposed (e.g., [1, 38]).

Combination of BPMN with i* and with KAOS has been addressed in [21] and [22], respectively. Although derivation of business process models from goal models has been addressed in [8, 15], challenges and problems such as insufficient concept mapping have been found [9].

Some works have presented ways to extend business process models with information related to non-functional requirements and goals. For example, service-level agreement information has been interwoven in business process models in [14], combination of variability analysis and non-functional requirements to drive the configuration of a business process is presented in [40], systematic use of soft-goals in process design was addressed in [42], and value-oriented process modelling has been discussed in [46]. Examples of works that have proposed explicit specification of goals in business process models are [27, 28, 35]. A review of different approaches for business process modelling can be found in [2, 23].

With regard to the extension of goal models with business process characteristics, i* diagrams were extended in [26] with details such as sequence constraints and event happening. Similar approaches have been presented in [8, 14]. i* diagrams have also been used to identify business processes [29] and to represent business process goals [6]. Although Map models have been used to model business processes (e.g., [32]), they do not include important information such as business process participants.

In summary, much research has dealt with the combination of goal and business process modelling, focusing on improving the techniques with details of the other and aligning them. This shows the relevance of their combination and that the techniques are not completely equivalent. However, no work has discussed and thus justified under what circumstances (1) both techniques can be considered equivalent and (2) a technique could be more suitable. In addition, a complete set of guidelines for derivation of goal models from business process models has not been provided yet.

Last but not least, some works (e.g., [16, 18, 43]) have discussed the selection of approaches for elicitation of system requirements. However, they have not analysed business process-based approaches thus neither compared them with goal-based ones.

3 Derivation of Goal Models from Business Process Models

This section presents how goal models in the form of goal trees can be derived from business process models. For this purpose, a set of preliminary concepts is introduced and a set of guidelines is provided.

3.1 Preliminary Concepts for Derivation of Goal Trees

Derivation of goal trees from business process models is based on several concepts. The concepts also aim to facilitate the explanation and understanding of the derivation process.

A **goal tree** consists of operational goals that are decomposed into other goals or tasks by means of AND and OR decompositions. A task is an atomic activity that is performed to fulfil a goal. The contributions of other goals or tasks are necessary to fulfil an operation goal. The semantics of an AND decomposition is that all the descendant elements have to be fulfilled (for goals) or performed (for tasks) in order to fulfil the decomposed goal. For an OR decomposition, the decomposed goal will be fulfilled when some of the descendant elements are fulfilled or performed. Therefore, OR decompositions depict alternative ways to fulfil a goal.

Several concepts have been defined to specify the guidelines for derivation of a goal tree from patterns of a business process model. These concepts might be complicated, but they are necessary to simplify the explanation of the guidelines. Fig. 3 shows some patterns modelled with BPMN that are used to explain the concepts.

The **basic flow** of a business process model is the set of elements that are executed in all the instances of the business process. In Fig. 3, the basic flow of BP1 is the set of elements {1, 2, 3, 5, 6, 7, 10, 13}.

An **alternative flow** in a business process model is a set of flow objects that is not part of the basic flow of the model and does not have more than one connection to another flow (regardless whether the flow is basic or alternative). In Fig. 3, the alternative flows of BP1 are the sets of elements {4}, {8}, {9} and {11, 12}. The set {9, 11, 12} is not an alternative flow because it would have two connections with the basic flow (9 and 12 with 10).

A **loop** in a business process model is an iteration of a sequence of elements of the model. In Fig. 3, the sequence of elements {16, 15} is a loop in BP2.

A **loop with alternative executions** in a business process model is a loop that contains elements that are part of the basic flow of the model as well as elements that are not. In Fig. 3, the loop {20, 21, 19} in BP3 is a loop with alternative executions.

An **alternative execution of a loop** in a business process model is each possible execution of a loop with alternative executions. The sequence of elements of the loop that are part of the basic flow of the model is an alternative execution of the loop too. In Fig. 3, the sequences of elements {19, 20} and {21, 19, 20} in BP3 are the alternative executions of the loop.

A **branching place** of a business process model is a place in the model where:

- an alternative flow begins, and;
- some alternative flow that begins from the place is not part of a loop whose end condition is checked in the place.

In Fig. 3, the branching places of BP1 are (3), (7) and (11). In BP4, (25) is a branching place too. However, place (20) in BP3 is not a branching place because it does not fulfil the second condition.

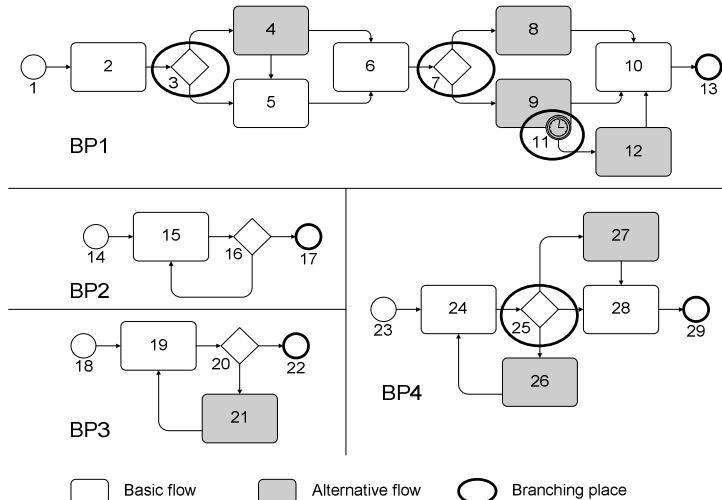


Fig. 3. Patterns in business process models

3.2 Guidelines for Derivation of Goal Trees from Business Process Models

Possibility of derivation of goal models from business process models was discussed and initially justified in Section 2.1 on the basis of the implicit (or explicit, depending on the notation) existence and modelling of operational goals in a business process model. This section presents the guidelines for derivation of goal trees.

We have defined these guidelines from the analysis of several, different BPDs, and also taking into account the structure of goal trees. The guidelines have been divided into four groups: derivation, refinement, contribution, and completion guidelines. For definition of the guidelines, BPMN terminology is used.

Derivation guidelines allow goals and tasks to be defined and named. Refinement guidelines allow the type of decomposition of a goal to be determined. Contribution guidelines allow contributions of goals and tasks to the fulfilment of other goals to be determined. Finally, completion guidelines allow a goal tree to be finished.

The contribution guidelines and the refinement guidelines are applied together. For example, the refinement guideline R.1 needs a contribution guideline (guideline C.1) in order to define the descendant elements of the goal that is refined.

Table 1 shows a summary of the guidelines. It presents the mapping of BPD elements and patterns into elements of a goal tree, as well as the elements of a goal tree and the type of decomposition that contribute to the fulfilment of a goal. Table 1 also provides the rationale of the guidelines implicitly. For example, a branching place in a business process model represents a goal that must be fulfilled in the process and can be fulfilled in different ways (i.e., by executing different branches).

Table 1. Summary of guidelines to derive a goal tree from a BPD

BPD element	Element of a goal tree	Decomposition	Descendent element
BPD	Goal	AND	Goals and tasks that do not contribute to another goal in the goal tree
Sub-process	Goal	-	-
Task	Task	-	-
Event with a trigger	Task	-	-
Loop with no alternative executions	Goal	AND	Goals and tasks derived from the BPD elements of the loop
Loop with alternative executions	Goal	OR	Goals derived from the alternative executions of the loop
Alternative execution of a loop	Goal	AND	Goals and tasks derived from the BPD elements of the alternative execution
Branching place	Goal	OR	Goals derived from the branches that follow the branching place
Branch that follows a branching place	Goal	AND	Goals and tasks derived from the BPD elements of the branch
Data object	Goal	AND	Goals and tasks derived from BPD elements that change the state of the data object and are not in a loop Goals derived from loops that change the state of the data object Goals derived from other data objects that are related to the data object by means of an inclusive aggregation relation

As an example, Fig. 4 shows the goal tree derived from the BPD in Fig. 1. The goal tree can be considered similar to a Tropos [4] or KAOS goal model [45]. In relation to this fact, a combination of the i^* notation for modelling of goals and tasks and of the structure of the KAOS goal model is used in the goal tree.

Table 2 shows the guidelines that have been applied to derive the goal tree in Fig. 4. For each element of the goal tree, the guidelines applied for its derivation, refinement, and contribution are specified. It must be noted that completion guidelines are not applied in this example.

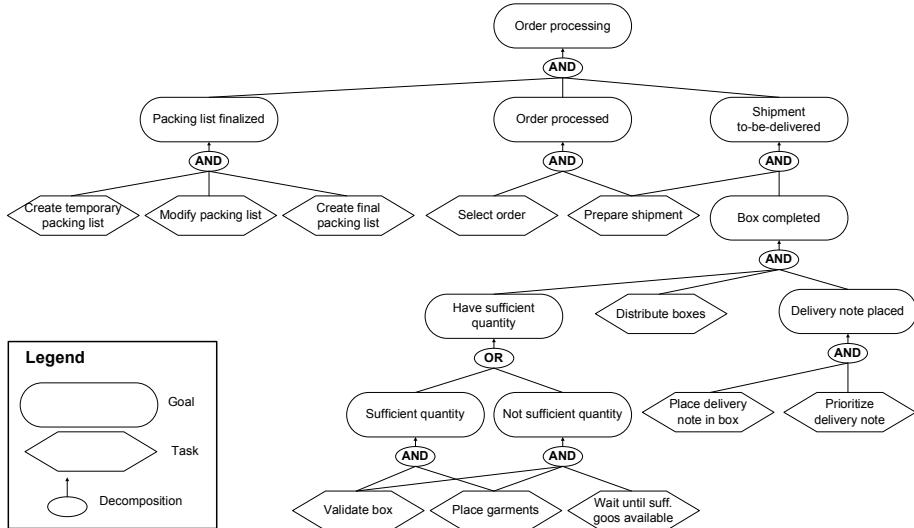
The next subsections present the guidelines of each group defined.

3.2.1 Derivation Guidelines

D.1 (BPDs). A BPD depicts a goal that corresponds to the root of a goal tree and is fulfilled when the business process ends. The name of the goal in the goal tree is the same as the name of the BPD.

D.2 (sub-processes). A sub-process in a BPD depicts a goal in a goal tree that is fulfilled when the sub-process ends. The name of the goal in the goal tree is the same as the name of the sub-process in the BPD.

D.3 (tasks). A task in a BPD depicts a task in a goal tree. The name of the task in the goal tree is the same as the name of the task in the BPD.

**Fig. 4.** Example of goal tree

D.4 (events). An event with a trigger in a BPD depicts a task in a goal tree (except link triggers, which are only used to link BPDs). The name of the task in the goal tree will depend on the criterion of the creator, but it has to refer to the event type (start, intermediate, final) and the event trigger (message, timer, cancel...).

D.5 (loops). A loop in a BPD depicts a goal in a goal tree that is fulfilled when the loop ends. The name of the goal will depend on the criterion of the creator, but it has to refer to the condition that is fulfilled when the loop ends.

D.6 (alternative executions of a loop). An alternative execution of a loop in a BPD depicts a goal in a goal tree that is fulfilled when the alternative execution is executed. The name of the goal will depend on the criterion of the creator.

Table 2. Guidelines applied to derive the goal tree of Fig. 4

Element of the goal tree	Guidelines
Order processing	D.1 / R.1 / C.9
Packing list finalized, Order processed, Delivery note placed	D.9 / R.1 / C.6
Create temporary packing list, Modify packing list, Create final packing list, Select order, Prepare shipment, Validate box, Place garments, Distribute boxes, Place delivery note in box, Prioritize delivery note	D.3 / - / -
Shipment to-be-delivered	D.9 / R.1 / C.6, C.8
Box completed	D.9 / R.1 / C.6, C.7, C.8
Have sufficient quantity	D.5 / R.2 / C.2
Sufficient quantity, Not sufficient quantity	D.6 / R.1 / C.3
Wait until sufficient goods are available	D.4 / - / -

D.7 (branching places). A branching place in a BPD depicts a goal in a goal tree that is fulfilled when all the branches that follow the branching place end or merge into the basic flow. The name of the goal will depend on the criterion of the creator.

D.8 (branches that follow a branching place). A branch in a BPD that follows a branching place depicts a goal that is fulfilled when the branch ends or merges into the basic flow. The name of the goal will depend on the criterion of the creator.

D.9 (data objects). A data object in a BPD whose state changes during the execution of the business process depicts a goal in a goal tree that is fulfilled when the data object reaches the last of its states in the BPD. The name of the goal is the name of the data object in the BPD followed by the last state that the data object reaches.

3.2.2 Refinement Guidelines

R.1 (BPDs, loops with no alternative executions, alternative executions of a loop, branches that follow a branching place, and data objects). A goal that is defined from a BPD, a loop with no alternative executions, an alternative execution of a loop, a branch that follows a branching place and whose first flow object belongs to an alternative flow, or a data object whose state changes during the execution of a business process, is refined in a goal tree by means of an AND decomposition.

R.2 (loops with alternative execution and branching places). A goal that is defined from a loop with alternative executions or a branching place is refined in a goal tree by means of an OR decomposition.

3.2.3 Contribution Guidelines

C.1 (elements of a loop with no alternative executions). The goals and tasks that are derived from the elements that are executed in a loop with no alternative executions contribute to the fulfilment of the goal of the loop in a goal tree.

C.2 (alternative executions of a loop). The goals that are derived from the alternative executions of a loop contribute to the fulfilment of the goal of the loop in a goal tree.

C.3 (elements of an alternative execution of a loop). The goals and tasks that are derived from the elements that are executed in an alternative execution of a loop contribute to the fulfilment of the goal of the alternative execution in a goal tree.

C.4 (branches that follow a branching place). The goals that are derived from the branches that follow a branching place contribute to the fulfilment of the goal of the branching place in a goal tree.

C.5 (elements of a branch that follows a branching place). The goals and tasks that are derived from the elements of a branch that follows a branching place and whose first flow object belongs to an alternative flow contribute to the fulfilment of the goal of the branch in a goal tree.

C.6 (data objects). The goals and tasks that are derived from tasks and sub-processes of a BPD, are not executed in a loop, and change the state of a data object contribute to the fulfilment of the goal of the data object in a goal tree.

C.7 (data objects in loops). The goals that are derived from loops whose execution changes the state of a data object contribute to the fulfilment of the goal of the data object in a goal tree.

C.8 (inclusive aggregation relations between data objects). The goals that are derived from a data object that is related to another data object in the domain data model by means of an inclusive aggregation relation (component data object) contribute to the fulfilment of the goal of the latter data object (composed data object) if defined in a goal tree.

C.9 (goals and tasks with no contribution). The goals or tasks in a goal tree that do not contribute to the fulfilment of some goal contribute to the fulfilment of the root of the goal tree.

3.2.4 Completion Guidelines

T.1 (goals with no descendants). The goals that do not have descendants in a goal tree and that have not been derived from a sub-process are changed into tasks.

T.2 (goals with only one descendant). The goals that have only one descendant are removed from a goal tree. The descendant will contribute to the fulfilment of those goals to which the parent goal contributes in the goal tree.

4 Discussion

Once the background, the guidelines for derivation of goal models, and an example of the correspondence between business process models and goal models have been presented in the previous sections, this section discusses the implications that this correspondence has in RE in general and how it is related to other works.

We have divided this section into four subsections to discuss (1) the correspondence between goal models and business process models, (2) when (only) goal models should be used, (3) when (only) business process models should be used, and (4) when both types of models should be combined.

Before presenting each subsection, it must be indicated that selection of goal modelling and/or business process modelling depends on more factors than those discussed in this section. For example, we have observed that many practitioners try to minimize combination of modelling techniques or that they may be reluctant to use a new technique [10]. Other authors have acknowledged similar issues (e.g., [16]). Therefore, these aspects must also be considered when adopting or proposing adoption of goal modelling and business process modelling for elicitation of system requirements. The discussion below does not take these issues into account, and simply present some recommendations based on our reflections and experience, both in academia [10] and in industry (e.g., [34]).

It must also be noted that business process models and goal models are similar and can be considered equivalent in some aspects, but not in all. Selection of one of the types of models should be justified and explained when modelling and analysing and organization or an IS, so that the decision and the rationale behind it are clear. As discussed below, the use of a type of model will depend on the part or aspects of the

application domain and of an IS with which system analysts and other stakeholders are mainly concerned.

The rest of this section presents each subsection defined, referring to other works when possible and considered relevant to support our arguments.

4.1 Correspondence between Goal Models and Business Process Models

As explained in the Introduction, goal-based RE approaches and business process-based RE approaches are initially and in general targeted at support of different aspects of an organization (objectives vs. activity). Nonetheless, we have shown how a goal model can be derived from a business process models, what implies that goal-oriented aspects are implicitly addressed when modelling business processes. As mentioned in Section 2.2, previous works have also studied the derivation of business process models from goal models. Although the goal models had to be extended with business process-oriented information, we think that these works support our argument about the fact that business process-oriented aspects are implicitly addressed when creating goal models.

Consequently, we think that these two perspectives should not be regarded as completely distinct. Past research on their combination has shown that they are complementary, and this paper shows that they can even be considered equivalent in some aspects (e.g., for modelling of operational aspects). Business process models allow specification of part of the information that is gathered and analysed in goal-oriented RE approaches, and goal models allow specification of part of the information that is gathered and analysed in business process-based RE approaches.

One interesting implication of this correspondence that we have found is related to compliance with safety standards in the development of critical systems. Two types of standards are distinguished commonly [20]: goal-based standards and prescriptive standards. The first type focuses on the definition of the objectives that the development of a safety-critical system must fulfil (e.g., “Requirements are specified”), whereas the second type focuses on the definition of the process, activities, and techniques to develop the system (e.g., determining how requirements must be specified by prescribing or recommending some specific techniques).

These two types of standards are usually considered to represent different perspectives for the development of safety-critical systems. However, and in line with the arguments presented in this paper, we think that they can be regarded as equivalent in some aspects. Indeed, compliance with any of the types of standards requires the definition and approval of a system lifecycle plan that meets the standards’ criteria. This plan basically corresponds to a business process for system analysis, development, verification and validation, maintenance, and decommission.

As also acknowledged in the system safety community (e.g., [20]), probably the main difference between the two types of the standards lies in the fact that goal-based standards usually present more abstract safety criteria. Consequently, they provide more flexibility with regard to the final decisions upon the process and techniques to use for developing a safety-critical system. This is in line with some of the main reasons for using goal models in RE [36, 45], and with the discussion below.

With regard to our past work, we have always believed that ISs must support the business processes of an organization, thus we have initially focused on business

process modelling in the approaches that we have developed and applied. However, we have also realised that there are some aspects such as the system purpose that cannot be always accurately captured in business process models. As a result, and for instance, we combine BPMN and Map and analyse the need of their use on the basis of the characteristics of a project.

4.2 When Should Goal Models Be Used?

We consider that goal models and business process models can be regarded as equivalent for modelling of some operational aspects of an organization. Nonetheless, there are situations in which goals models might be considered better suited for modelling of business requirements.

For situations in which an organization does not have a clear procedure defined, or even it does not exist, we considered that the use of goal models would be more adequate. First, designing and modelling business processes “from scratch” could be very difficult because employees would not be able to provide information about the procedures they follow. Consequently, their validation could also be hindered. Second, by modelling and analysing (strategic) goals, system analysts can at least try to guarantee that the system requirements meet organizational goals. Support for the operational aspects, once the strategic ones have been refined, would imply support for business process aspects (i.e., for organizational activity).

Finally, we consider that in situations in which no procedure exists, goal modelling facilitates variability analysis. We think that it is easier to model and analyse alternatives in goal models than directly model business processes, trying to define alternative paths without any rationale such as the possible alternative ways to fulfil a given goal. In addition, guidance can easily be found regarding analysis of alternatives in goal models (e.g., [45]).

Goal modelling can also be regarded as an advisable initial step that facilitates modelling of new business processes in these situations.

4.3 When Should Business Process Models Be Used?

We consider that there is a situation in which the use of only business process models is the most suitable option: development of an IS for an organization that has defined procedures and that mainly needs automation support for its current procedures. Since no fulfilment of new goals or big changes (apart from automation) would be required and expected, we consider that goal models would not be necessary. At least, this is what we have experienced and observed in practice [10, 34]. In many situations, we only use BPMN and do not combine it with Map. It can be argued that this type of projects are not very complex to deal with, but it is also true that, to our knowledge, this is probably the most frequent situation when developing an IS.

In relation to the approach proposed in [11], we now consider that derivation of goal models from business process models for elicitation of system requirements would not be necessary in situations in which automation is the main benefit expected from a new IS. Automation can directly be analysed in business process models, thus modelling of goals may correspond to an unnecessary effort. Furthermore, goal trees

derived from business process models can become very tangled (thus difficult to understand and manage) for complex business processes. Therefore, derivation and analysis of goal trees may not be advisable for these business processes. Studying possible improvements on the guidelines presented might mitigate this problem.

On the other hand, and in line with the discussion above, derivation of a goal tree might facilitate the analysis of alternative, new ways to execute a business process.

4.4 When Should Goal Models and Business Process Models Be Combined?

We consider that combination of goal models and of business process models is clearly justifiable and even necessary in situations in which organizational procedures are (more or less) well-defined, but an organization expects a change in them as a result of the development of an IS and the system must also support fulfilment of some strategic goal. This is the type of situation we addressed in [12, 34], in which combination of BPMN and Map was proposed. This situation and the proposed solution is also line with works such as [3, 19, 26].

On the one hand, combination of goal models and business process models allow all types and abstraction levels of goals of an organization and of an IS to be addressed. In our approach, strategic goals are modelled and analysed on the basis of Map, whereas operational goals are modelled and analysed on the basis of BPMN.

On the other hand, Map complements BPMN by allowing system analysts to analyse the purpose of an IS on the basis of the strategic goals of an organization. BPMN complements Map by allowing system analysts to model details of organizational activity that cannot be modelled with the goal-oriented RE approach or whose modelling presents limitations.

In general and in summary, the combination of goal models and business process model allows analysis of the “why” (goals) and the “what” and “how” (business processes) aspects of the business requirements for an IS.

5 Conclusions and Future Work

This paper has discussed the use and correspondence of goal models and business process models for elicitation of system requirements of an IS. The discussion has been mainly driven by the possibility of deriving a goal model from a business process models. Such derivation is based on a set of 22 guidelines for mapping of patterns and elements of a business process model into a goal tree. The guidelines allow derivation and refinement of goal tree elements, determination of the elements to which another contributes, and completion of a goal tree. They show how both types of models can be regarded as equivalent in some operational aspects.

Although goal models and business process models can complement each other and can be considered equivalent in some operational aspects, we consider that there are situations in which their combination is not necessary or use of only one technique is more suitable. Goal models should be used when dealing with new situations in an organization, with strategic goals, or with variability, whereas business process models should be used when an IS is mainly aimed at supporting and automating

existing, running activity of an organization. Both types of models should be combined if both strategic and known operational issues had to be addressed.

As future work, we want to validate the guidelines presented by analysing their support to workflow patterns [44], and to analyse in detail the quality of the goal models derived. We also want to gain further insights into the use of goal models and business process models in practice. Finally, we would like to analyse how this paper relates to others on the selection of RE approaches (e.g., [18, 43]).

Acknowledgments. The research leading to this paper has received funding from the Research Council of Norway under the project Certus SFI (Project No. 203461/030), the FP7 programme under the grant agreement n° 289011 (OPENCOSS), the Spanish Government under the project PROS-Req TIN2010-19130-C02-02, the Valencia Regional Government under the project ORCA PROMETEO/2009/015, and ERDF.

References

1. Antón, A.: Goal-Based Requirements Analysis. In: ICRE (1996)
2. Bider, I.: Choosing Approach to Business Process Modeling. *Journal of Conceptual Modeling* 34 (2005)
3. Bleistein, S., et al.: B-SCP: A requirements analysis framework for validating strategic alignment of organizational IT. *Information and Software Technology* 48(9), 846–868 (2006)
4. Bresciani, P., et al.: Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems* 8(3), 203–236 (2004)
5. Bubenko, J., Persson, A., Stirna, J.: EKD User Guide (2001)
6. Cardoso, E.C.S., Almeida, J.P.A., Guizzardi, G., Guizzardi, R.S.S.: Eliciting Goals for Business Process Models with Non-Functional Requirements Catalogues. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009. LNBP, vol. 29, pp. 33–45. Springer, Heidelberg (2009)
7. Cortes-Cornax, M., Matei, A., Letier, E., Dupuy-Chessa, S., Rieu, D.: Intentional Fragments: Bridging the Gap between Organizational and Intentional Levels in Business Processes. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 110–127. Springer, Heidelberg (2012)
8. Decreus, K., Poels, G.: A Goal-Oriented Requirements Engineering Method for Business Processes. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBP, vol. 72, pp. 29–43. Springer, Heidelberg (2011)
9. Decreus, K., Snoeck, M., Poels, G.: Practical Challenges for Methods Transforming i* Goal Models into Business Process Models. In: RE 2009 (2009)
10. de la Vara, J.L.: Business process-based requirements specification and object-oriented conceptual modelling of information systems. PhD thesis, Univ. Pol. de Valencia (2011)
11. de la Vara, J.L., Sánchez, J.: Business process-driven requirements engineering: a goal-based approach. In: BPMDS 2007 (2007)
12. de la Vara, J.L., Sánchez, J., Pastor, Ó.: Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 213–227. Springer, Heidelberg (2008)
13. Eriksson, H., Penker, M.: Business Modeling with UML: Busines Patterns at Work. Wiley (2000)

14. Frankova, G., et al.: Deriving business processes with service level agreements from early requirements. *Journal of Systems and Software* 84(8), 1351–1363 (2011)
15. Ghose, A.K., Narendra, N.C., Ponnalagu, K., Panda, A., Gohad, A.: Goal-Driven Business Process Derivation. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 467–476. Springer, Heidelberg (2011)
16. Hickey, A.M., Davis, A.M.: Elicitation Technique Selection. In: RE 2003 (2003)
17. Indulska, M., Green, P., Recker, J., Rosemann, M.: Business Process Modeling: Perceived Benefits. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 458–471. Springer, Heidelberg (2009)
18. Jiang, L., et al.: A methodology for the selection of requirements engineering techniques. *Software and Systems Modeling* 7(3), 303–328 (2008)
19. Kavakli, V., Loucopulos, P.: Goal-Driven Business Process Analysis Application in Electricity Deregulation. *Information Systems* 24(3), 187–207 (1999)
20. Kelly, T., McDermid, J., Weaver, R.: Goal-Based Safety Standards: Opportunities and Challenges. In: SSS (2005)
21. Koliadis, G., Vranesovic, A., Bhuiyan, M.A., Krishna, A., Ghose, A.K.: Combining *i** and BPMN for Business Process Model Lifecycle Management. In: Eder, J., Dustdar, S. (eds.) BPM 2006 Workshops. LNCS, vol. 4103, pp. 416–427. Springer, Heidelberg (2006)
22. Koliadis, G., Ghose, A.: Relating Business Process Models to Goal-Oriented Requirements Models in KAOS. In: Hoffmann, A., Kang, B.-H., Richards, D., Tsumoto, S. (eds.) PKAW 2006. LNCS (LNAI), vol. 4303, pp. 25–39. Springer, Heidelberg (2006)
23. Krogstie, J.: Perspectives to Process Modeling – A historical overview. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (eds.) BPMDS 2012 and EMMSAD 2012. LNBP, vol. 113, pp. 315–330. Springer, Heidelberg (2012)
24. Kueng, P., Kawalek, P.: Goal-based Business Process models. *Business Process Management Journal* 3(1), 17–38 (1997)
25. Lauesen, S.: Software Requirements: Styles and Techniques. Addison-Wesley (2002)
26. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)
27. Lin, Y., Sølvberg, A.: Goal Annotation of Process Models for Semantic Enrichment of Process Knowledge. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 355–369. Springer, Heidelberg (2007)
28. Markovic, I., Kowalkiewicz, M.: Linking Business Goals to Process Models in Semantic Business Process Modeling. In: EDOC 2008 (2008)
29. Mazón, J.-N., Pardillo, J., Trujillo, J.: A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses. In: Hainaut, J.-L., et al. (eds.) ER Workshops 2007. LNCS, vol. 4802, pp. 255–264. Springer, Heidelberg (2007)
30. McKeen, J., Smith, H.A.: Making IT Happen. Wiley (2003)
31. Nicolás, J., Toval, A.: On the generation of requirements specifications from software engineering models. *Information and Software Technology* 51(9), 1291–1307 (2009)
32. Nurcan, S., et al.: A strategy driven business process modelling approach. *Business Process Management Journal* 11(6), 628–649 (2005)
33. OMG: Business Process Model and Notation (BPMN), Version 1.2 (2009)
34. OPENCROSS project, <http://www.opencross-project.eu>
35. Ould, M.: Business processes: modelling and analysis for re-engineering (1995)
36. Pohl, K.: Requirements Engineering. Springer (2010)

37. Pourshahid, A., et al.: Business process management with the user requirements notation. *Electronics Commerce Research* 9(4), 269–316 (2009)
38. Rolland, C., Souveyet, C., Ben Achour, C.: Guiding Goal Modeling Using Scenarios. *IEEE Transactions on Software Engineering* 24(12), 1055–1071 (1998)
39. Rolland, C.: Capturing System Intentionality with Maps. In: *Conceptual Modelling in Information Systems Engineering*, pp. 141–158. Springer (2007)
40. Santos, E., Pimentel, J., Castro, J., Sánchez, J., Pastor, O.: Configuring the Variability of Business Process Models Using Non-Functional Requirements. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) *BPMDS 2010 and EMMSAD 2010. LNBP*, vol. 50, pp. 274–286. Springer, Heidelberg (2010)
41. Scheer, A.W.: *ARIS - Business Process Modeling*, 3rd edn. Springer (2000)
42. Soffer, P., Wand, Y.: On the notion of soft-goals in business process modelling. *Business Process Management Journal* 11(6), 663–679 (2005)
43. Tsumaki, T., Tamai, T.: Framework for Matching Requirements Elicitation Techniques to Project Characteristics. *Softw. Process: Improvement and Practice* 11(5), 505–519 (2006)
44. van der Aalst, W., et al.: Workflow patterns. *Distrib. and Parallel Databases* 14(1), 5–51 (2003)
45. van Lamsweerde, A.: *Requirements Engineering*. Wiley (2009)
46. vom Brocke, J., Recker, J., Mendling, J.: Value-oriented process modeling: integrating financial perspectives into business process re-design. *Business Process Management Journal* 16(2), 333–356 (2010)
47. Yu, E.: *Modelling Strategic Relationships for Process Reengineering*. PhD Thesis, University of Toronto (1995)

Multi-level Autonomic Business Process Management

Karolyne Oliveira¹, Jaelson Castro¹, Sergio España², and Oscar Pastor²

¹ Universidade Federal de Pernambuco—UFPE, Recife, PE 50 740-560, Brazil

² Centro de Investigación en Métodos de Producción de Software,

Universitat Politècnica de València, Valencia, Spain

{kmao, jbc}@cin.ufpe.br,

{sergio.espana, opastor}@pros.upv.es

Abstract. Nowadays, business processes are becoming increasingly complex and heterogeneous. Autonomic Computing principles can reduce this complexity by autonomously managing the software systems and the running processes, their states and evolution. Business Processes that are able to be self-managed are referred to as Autonomic Business Processes (ABP). However, a key challenge is to keep the models of such ABP understandable and expressive in increasingly complex scenarios. This paper discusses the design aspects of an autonomic business process management system able to self-manage processes based on operational adaptation. The goal is to minimize human intervention during the process definition and execution phases. This novel approach, named MABUP, provides four well-defined levels of abstraction to express business and operational knowledge and to guide the management activity; namely, Organizational Level, Technological Level, Operational Level and Service Level. A real example is used to illustrate our proposal.

Keywords: Autonomic business process models, workflow management, self awareness, context awareness.

1 Introduction

System components and software have been evolving to deal with the increasing complexity of stakeholder needs. Automating the management of computing resources is a major challenge.

Some recent works have proposed the use of Autonomic Computing (AC) concepts [1] to help to effectively manage enterprise systems and applications [3]. Indeed, a major application area for autonomic computing is aimed at freeing system administrators from the details of system operation and maintenance, improving robustness of systems and decreasing the total cost of ownership [2, 6].

Autonomic Computing Systems (ACS) are able to: (i) Self-configure; (ii) Self-heal; (iii) Self-optimize and, (iv) Self-protect. The secondary properties of autonomic systems are: (i) Self-awareness: An autonomic system requires to know itself; (ii) Context-awareness: An autonomic system should be aware of its execution environment by

exposing itself and discovering other systems in the environment; (iii) Openness: An autonomic system should be able to function in a heterogeneous environment and be implemented on open standards and protocols; (iv) Anticipatory: One critical property from the perspective of the users is that an autonomic system should be able to anticipate its needs and behaviors to act accordingly, while keeping its complexity hidden [10].

However, the vision of autonomic computing should be not restricted to the area of system administration, management and maintenance. For example, it can also be applied to the area of process-aware information systems to effectively and efficiently deal with changes in several aspects of these applications.

Business Processes and Business Process Management (BPM) are essential in many modern enterprises. They constitute **organizational** and **operational knowledge** and often perpetuate independently of changes in the personnel or the infrastructure [4]. Autonomic computing principles can also be adapted to help organizations survive in dynamic business scenarios.

A Process that is compliant with AC principles is referred to as Autonomic Business Process (ABP) [9] [12]. ABPs (a.k.a. autonomic workflows) must have the capability to adjust to **environment variations** (context). If one **component service node** of an ABP becomes unavailable, a mechanism is needed to ensure a business process execution is not interrupted [8] [18]. ABP differs from traditional workflow in the fact that it relies on autonomic techniques to manage adjustments during its execution. Therefore, it enables dynamic and automatic configuration of its definition, activities and resources. It also allows self-optimization and self-healing. Furthermore, autonomic workflows must have intelligence to analyze situations and deduce adaptations at run-time.

This work investigates the application of autonomic computing principles to business processes management. Our goal is to help organizations survive in dynamic environments. More specifically we want to face an open challenge in the area, which is to promote modularization and separation of concerns (SoC) in ABP Models [14].

This paper proposes a framework that exploits the high variability in service-oriented system environments by using models of the system context and by providing autonomic adaptations which rely on operationalizations of Non-Functional Requirements (NFR). This framework guides its adaptation according to a Multi-level Autonomic Business Process named MABUP, whose modeling process is presented in [9]. The benefits are manifold and include addressing scalability problems and improving the understandability of ABP in complex scenarios [11].

A MAPE cycle (Monitor-Analyze-Plan-Execute) is used, considering both the system (self) and the instrumented BPM (context).

This paper is structured as follows. Section 2 presents the background and related works. In Section 3, we introduce a running example. Section 4 proposes a multi-level autonomic business process management approach. Lastly, Section 5 discusses the proposal and outlines further research.

2 Related Works

Strohmaier and Yu in [13] have presented the first attempt to apply autonomic computing principles to workflow management. Their work shows that certain levels of autonomy can already be achieved with available techniques and introduces the concepts of the different degree of “autonomy” in workflow systems. However, no novel technique is proposed.

In order to deal with autonomic features, an interesting issue emerged when composing applications: the ability to bind and re-bind abstract activities to concrete services at run-time. Several researchers have addressed this issue. Lee et. al. [7] discussed managing run-time adaptations in long-running scientific workflows. In their work, adaptations have been described in terms of the functional decomposition of autonomic managers into monitoring, analysis, planning and execution components. Mosincat et al. [8] proposed fault tolerant execution of BPEL processes executing dynamic binding of services, performing a process transformation before the execution, for using a selection component at run-time. In Haupt et. al. [5], the workflow and the services comprising it are treated as managed resources controlled by hierarchically organized autonomic managers. Their work attempts to treat the complexity problem of autonomic workflows by using hierarchical workflow but, it does not explore any modularity technique neither how these different levels can guide different kinds of adaptations.

The modularization helps to treat the complexity of large-size models [11]. Once the management of business process models can be realized at different levels of granularity, there is a tradeoff between monitoring granularity and diagnostic precision. The finest level of monitoring granularity is at the functional level where all leaf level tasks are monitored. The disadvantage of fine-grained level monitoring is high monitoring and diagnostic overhead. Coarser levels of granularity only monitor higher-level business goals. In these cases, less complete and high-level monitoring data is generated, leading to multiple competing diagnoses. Both monitoring and diagnostic overheads are lower. The disadvantage is that if requirements denials are found, multiple diagnoses are returned, each pinpointing possible failures [17].

Generally, self-adaptive software is a closed-loop system with feedback from the self and the context. The Autonomic Computer System’s building block, named autonomic manager, constantly interacts with managed element in order to maintain its equilibrium in face of incoming perturbations. The MAPE cycle (Monitor-Analyze-Plan-Execute), represented in Figure 1, is an implementation of the classical feedback control technique [6].

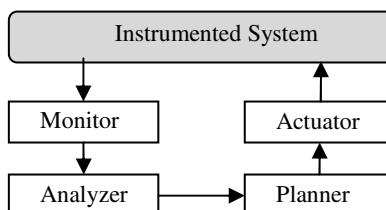


Fig. 1. Closed Loop Control Mechanism

Monitoring the execution context represents another important aspect to implement an autonomic behavior and for reacting to events. However, even if, they exploit workflows to design applications and to specify causal constraints among activities and pages, the major works do not use contextual information in the autonomic business process management system [15].

Furthermore, these works do not explore another crucial issue that is the expressiveness of the operational knowledge in the business process. This problem is related to how the metrics impact the in the management of business process and how to offer understandability of autonomic behavior.

The analysis of the related work shows that there is a lot of interest in the realization of systems and approaches able to deal, dynamically and automatically, with composition, binding, failures and other aspects of definition and execution of composed processes. However, these works do not explore the modularization of business process that helps to treat the management of autonomic business process in complex scenarios and expressiveness of autonomic features in business process models.

3 Running Example

A running example will be used to illustrate our approach. The example consists of a large system that requires to be managed in an autonomic manner. Specifically we treated the characteristic of Self-Optimization. We examine the CAGED (General Register of Employed and Unemployed), a project under the Ministry of Labour and Employment of Brazil (MTE) and governed by law 4923/65. It supports the submission of monthly declarations of change of company's employees due to dismissal or admittance (CAGED movements). The deadline for submission is the 7th day of every month. The data submitted are related to the previous month (i.e. its competence). The declarations are processed to generate operational and statistical data for the ministry of labor and employment.

Considering these characteristics, the process is started when the MTE opens the competence of CAGED reception. In this case, the declarants can send their CAGED declaration. The submit process a CAGED declaration is composed of a selection (including filling) of a CAGED file (detect the movements), sending this file and receiving a CAGED Receipt. During the reception, MTE can decide to start the processing of the files that provide operational data for MTE. Depending on timing constraints, such as holidays or some other guidance, MTE staff closes competences of reception. After all checks in CAGED reception and processing, operational staff closes the reception processing and starts statistical processing.

Having in mind that the CAGED submission is an activity that is critical to the success of CAGED process as a whole, we explain the main points that must be considered in its management. The capture operation of a CAGED file can be performed in different manner: (i) Generate CAGED File, that in general is executed through payroll systems of accounting firms which generate a CAGED declaration (in a predefined layout) without MTE analysis and signature; (ii) Generate Analyzed CAGED File, that is executed by declarants through a MTE offline system that

generate a CAGED File with analysis and signature; and (iii) Generate Short Analyzed CAGED File, that is performed by declarants to generate and send CAGED declarations with a maximum of 36 movements.

The CAGED Declaration can be generated in two ways: CAGED File without signature of analysis or an Analyzed and Signed CAGED File. The first one can be analyzed through the activity of Generate Analyzed CAGED File or to be analyzed and sent by a service. An analyzed CAGED file, the second one, is sent without additional analysis; in this case, two manners are possible: CAGED File with less than 1500kb can be sent through a service that only verifies MTE signature. CAGED Files with more than 1500kb must be sent through a desktop tool that optimizes its transmission.

The previous tasks are executed by the declarants that send their CAGED file to MTE. A MTE reception service is available to receive and store these files and triggers the generation of a CAGED Receipt to the declarants.

CAGED is complex system that is hard to be managed. Some qualities attributes must be assured according to its SLA such as availability, response time, processing time, etc; and all these with less human intervention. In this sense, business process models must be able to adequately represent: (i) Mission-critical Activities; (ii) The way the activities are executed and monitored; (iii) Which quality attributes and environmental context must be monitored and to what autonomic features; (iv) How systems needs to be adapted in case of some quality attributes deviation.

4 Multi Level Autonomic Business Process Management

To provide autonomic business process, we considered the use of modularization and separation of concerns to represent the different features. In Figure 2, we depict an overview of our approach to the management of a multi-level business process model that includes two main stages: (i) The modeling phase exploits the modeling of four abstraction levels: Organizational Level, Technological Level, Operational Level and Service Level; (ii) The management phase includes a MAPE module that uses the modeled autonomic business process and the metrics provided by the systems to guide the adaptation.

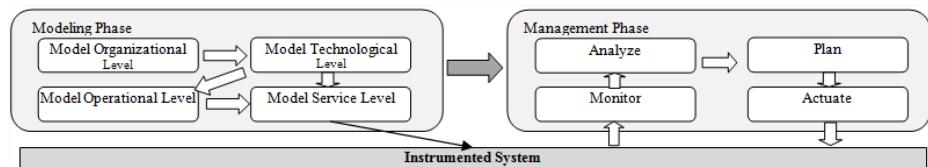


Fig. 2. Overview of the modeling and management phases of MABUP

In the sequel we present our Multi-Level Autonomic Business Process approach, named MABUP, which consists of well-defined abstraction level and a closed-loop mechanism to provide system adaptation (see Figure 2). As mentioned, our framework considers in the management of the processes both context information and quality attributes of the system.

4.1 Modeling Phase

In this section we present the four well-defined abstraction levels of MABUP. In the modeling phase, we used a combination of two languages; namely Communicative Event Diagram and BPMN (see Figure 3). Due to lack of space, we depict a simplified version of the models; for instance, we do not capture all context variations required for self-configuration, self-healing and self-protection. Hence, this paper focuses on the self-optimization feature of the “Declarant submits a declaration” critical communicative event.

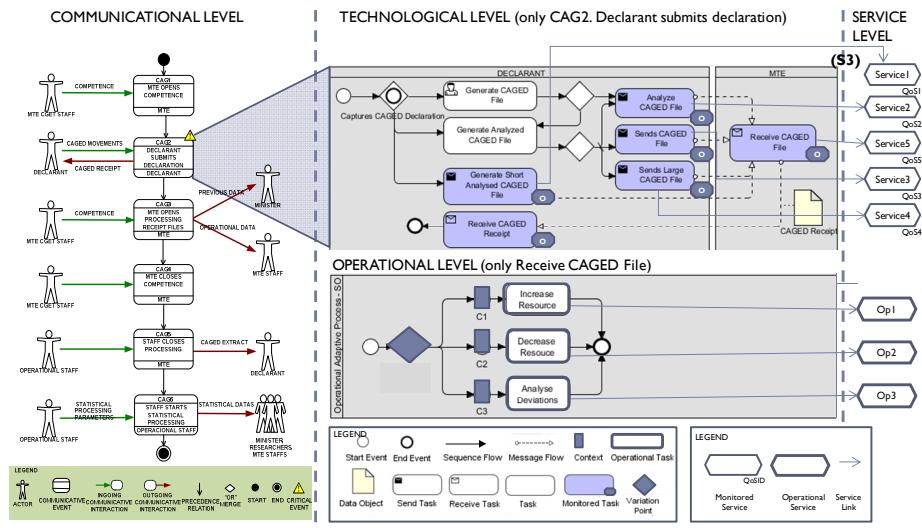


Fig. 3. Organizational, Technological, Operational and Service Levels

Model Organizational Level

The Organizational Level is an important abstraction level in our approach since it makes business process models more immune to changes in the support technologies by focusing on the essentials of the business behavior. It supports the modularization of Autonomic Business Process by providing a level which presents critical activities that must be refined and monitored.

According to the principles of Communication Analysis, Communicative Event Diagram provides a notation to specify the Organizational Level of a business process [1]. In this notation, there is a special primitive for process modelling, named communicative event. It represents the triggering of an activity that receives an incoming message, processes it and provides an output. Hence, it represents the organizational behavior resulting from a given change in world (subject system), intended to account for that change by gathering information about it. A communicative event is a set of actions related to information (acquisition, storage, processing, retrieval, and/or distribution), that are carried out in a complete and uninterrupted way, on the occasion of an external stimulus.

Communication Analysis offers guidelines to allow the identification of communicative events. The following modularity criteria are used to guide modeling [3]:

- Trigger unity, that presumes that the event occurs as a response to an external interaction by an actor that triggers organizational reaction;
- Communication unity, that describes that each and every event involves providing new meaningful information;
- Reaction unity, that outlines that an event triggers an Information System (IS) reaction, which is a composition of synchronous activities. Events are asynchronous among each other. This criterion defines a temporal encapsulation.

MABUP introduces the concept of Critical Event, a special kind of communicative event which is a mission-critical activity. Critical events must be refined to provide information about its behavior and indicate the sub-activities that are monitored according to autonomic features.

In this sense, Figure 3 presents the Organizational Level of Business Process of CAGED, where we highlight the critical event “CAG2 - Declarant submits declaration”. It is an activity that has as input all CAGED movements and, as an output, the receipt. Stakeholders highlight that it is critical to the success of the CAGED process.

Model Technological Level

Technological Level represents the refinement of a critical event processing to model important aspects that can impact software adaptation, such as:

- *Present different alternatives to perform an activity:* Some activities can be executed in different ways in a company. It is important to map these differences to analyze if they require special ways to be managed.

In our example, the generation of file can be performed in three different manners: (i) Generate CAGED File; (ii) Generate Analyzed CAGED File; and (iii) Generate Short Analyzed CAGED File. If one of these activities became unavailable, another alternative can be executed to guarantee the system operability until all processes return to an optimum state.

- *Indicate External dependences:* External dependences are important to be expressed as they can lead to interoperability problems. In that sense, interoperability demands human intervention coordination related to the efforts to ensure performance, scalability, correctness, or reliability of applications in the presence of concurrency and failures [2]. We consider that this calls for autonomic characteristics such as optimization, healing and protection. Furthermore, when a system relies on an external service, some kind of service level agreement (SLA) is required.

Figure 3 indicates that the “Receive CAGED Receipt” task is impacted if the “Receive CAGED File” task becomes unavailable or has a poor performance.

- *Highlight monitorable tasks:* Some events are too complex and have to be processed by different kind of components. Hence, some are monitorable and others not.

For example the “Generate analyzed CAGED File” task is executed outside the MTE domain by an offline desktop tool that is not expressed in SLA as a monitorable module. In our example the monitored tasks, such as “Receive CAGED File” are depicted in gray.

- *Define the autonomic characteristics and symptoms of monitorable tasks:* Indicate the autonomic principles that will be considered to monitor the tasks. The SLA document is a good source of information as it presents the quality attributes and their respectively values that must be assured.

In the running example, we only consider the self-optimization (O) principle to all monitorable tasks.

Model Operational Level

Operational level indicates the operational knowledge required to manage the process. Business analysts define the contextualization of the monitored task using their knowledge about the business domain to identify information that can affect the process and express the operational knowledge to manage it. The information is codified in terms of contexts, context variability and operational tasks.

The variability analysis is focused on context variability. Both variants and variation points are related with the contextual information. They indicate how the contexts variation can affect the autonomic actions. Operational Task expresses autonomic actions in the systems to assure the optimal state of the system and express a refinement of the autonomic part of a Monitored Task.

In our running example, we defined Self-Optimization (O) as the desired autonomic principle. As previously noted, it is related to the efficiency NFR, which in turn can be decomposed into others characteristics such as Response Time and Space Utilization. In this example, we deal only with Response Time attribute.

For instance, a variation point of the Receive CAGED File (see Table 1) task is associated with the response time. This variation point is a contextualization for the alternative ways how the adaptation actions are selected to respond to environment changes. The alternatives are represented as variants that are associated with the variation point. In the example, the variants are actions to control or regulate the response time deviation of the CAGED file reception. The tasks are associated with contexts describe in expressions that activate the presence of a variant in the variation point.

Given that in our running CAGED there is a deadline for declaration submission, it is important to measure the daily reception. As observed in Figure 4, the reception rate peaks during the first seven days of the month. This trend must be considered to allow the selection of a correct adaptation. Trend analysis methods [16] can help business analysts to predict future outcomes tracking variances in historical results.

In order to treat Response Time deviations that may be related to the performance of the “Receive CAGED File” activity, we defined in Figure 3 (Operational Level) three different tasks (operationalizations): (i) Increase resource; (ii) Decrease Resource; and (iii) Analyze deviation. The 3 contexts (C1, C2 and C3) present in Figure 3 (Operational Level) are defined in Table 1.

Table 1. Contextualization of Autonomic Business Process

Monitored Task	Contextualization			
	Variation Point	Variants	Context	Quality
Receive CAGED File	Response Time Deviation	Increase Reception Rate	C1: ReceptionTrendIsOK= true and LastThreeCycleIncreasing= true	Response Time >= 220ms
		Decrease Reception Rate	C2: ReceptionTrendIsOK= true and LastThreeCycleDecreasing= true	Response Time <= 110ms
		Deviation Reception Rate	C3: ReceptionTrendIsOK= false and LastThreeCycleIncreasing= true	Response Time >= 220ms

Considering the above, the operational level presents the new interconnected concepts in ABP models: variation point, variants, operational task and can be defined as follow:

$$OL = \{VP, Var, OT, Rel\{VPxVarxOT\}\}$$

where: OL is the Operational Level; VP is Variation Point; Var is Variant, OT is Operational Task and Rel expresses the relationship between them. OL is a conjunct of VP, Var and OT where VP is linked with Var and Var is related with OT.

Model Service Level

Both monitorable and operational tasks should be linked to system services. In a service-based mission-critical system, adaptation is an activity with the objective of delivering an acceptable/guaranteed service based on SLA (Service Level Agreement).

One of the key components in SLA is SLO (Service Level Objective) which specifies QoS (Quality of Service) metrics governing service delivery. Each SLA may include several SLOs, each corresponding to a single QoS parameter related to quality factors.

In the Service Level, the services linked with monitorable tasks are called monitorable services that should be checked according to the parameters presented in the SLO. Whereas the services linked with operational tasks are named as operational services that have the objective of returning the system to an optimal state in an autonomic manner. For example, the “Receive CAGED File” activity is linked to “Service5” that should assure a response time of 190 ms. The concepts used in this level are related as follow:

$$\text{MonitoredService} \subseteq \text{Service} \text{ and } \text{OperationalService} \subseteq \text{Service} \quad (\text{MonitoredService} \cap \text{AutonomicService} = \{\}) \wedge \text{MonitoredService} \cup \text{AutonomicService} \subseteq \text{Service}$$

That is, *MonitoredService* and *OperationalService* are a sub-conjunt of *Service* and are mutually exclusive because other services may exist (ie. a disjoint and incomplete inheritance of *Service*).

4.2 Management Phase

Monitor

The monitoring component collects indicators provided by the system from time to time (cycle). The monitor checks both context information and NFRs related to the system.

In our approach we assume the use of context sensors, which are computational entities that provide raw data about the operational environment presented in the instrumented BPM; and service quality attributes that must be assured. The monitor module verifies the processes at a well-defined level of modularity (Technological Level) and affects services-oriented systems with coarse grained adaptation at run-time.

As shown in Figure 5, the monitor module is divided in Context Monitor, QoS Monitor and Monitor Engine. The context monitor reads and processes the contextual data provided by the instrumented system. The QoS Monitor reads and processes the log database of all monitored services that support all monitored tasks and their respective metrics according to the SLO. The Monitor Engine processes the information provided by Context Monitor and QoS Monitor, collects the relevant data according MABUP model and passes it to the diagnostic component for analysis.

Algorithm 1 defines how the monitor process works, ie. it obtains the MABUP model, treats the monitored tasks, collects the metrics of each one and returns the data to be used by analyzer module.

Algorithm 1. Monitor algorithm

```

monitor(MABUPModel model)
1. MonitoredTaskValue[] mtvs
2. VariationPointValue vpv
3. VariationPointValue[] vpvs
4. for each tl • getTechnologicalLevels(model)
5.   do for each mti • getMonitoredTasks(tl)
6.     MonitoredService ms ← getMonitoredService(mt)
7.     OperationalLevel op • getOperationalLevel(mt)
8.     do for each vpj • getVariationPoints(op)
9.       vpv ← getVariationPointValues(mt, ms, vp)
10.      vpvs[j] ← vpv
11.    end for
12.    mtvs[i] ← getMonitoredTaskValues(mt, vpvs)
13.  end for
14.end for
15.return mtvs

```

The *getMonitoredTaskValues* method (line 12) localizes, though QoS Monitor, the metrics of the monitored service expressed in the Service Level based on the identification of the monitored task. It calculates the average of the metrics obtained in a cycle by all requested services that support the monitored task as follow:

$$f(s) = \frac{1}{n} \sum_{i=1}^n QoS(s_i)$$

where: f(s) the function to verify the QoS of a service s; n is the number of requested services in a cycle; QoS(s_i) is the metric obtained for a requested service.

Analyze

In our approach, this component has the objective to evaluate the metrics obtained by the monitor. Considering the measures obtained by the monitor, the diagnostic component checks the contexts expressed in the Business Process Model. In case of some deviation, the planner (section 4.2.3) selects the appropriate operational task that represents autonomic interventions in the system.

Considering the management of business processes, it is important to assure the service level agreement related to these processes. In this sense, the analyzer module verifies the monitored metrics in the business process model and in case of deviation above a predefined threshold the module verifies the operational contexts that affect the process and the variants related to these contexts. The analyzer module is divided in two components: Domain Assumption Verifier and Contextual Business Process Model Diagnosis.

The “Domain Assumption Verifier” component identifies all quality attributes related to the monitored task and in case of metric deviations the “Contextual Business Process Model Diagnosis” component analyzes the contexts values. Algorithm 2 is a simplified version of this process. According to lines 2-4 the “Domain Assumption Verifier” component reads all monitored task values obtained by the monitor module and verifies the QoS deviation. The isQoSDeviated method (line 3) reads the Service Level, gets the expected metrics and compare them against the obtained values. In the “Contextual Business Process Model Diagnosis” component, the getDeviatedVariant method (line 5) checks all Variation Points and evaluates the context and quality expected in each Variant. The Analyzer module returns all the deviated variants.

Algorithm 2. Diagnosis algorithm

```

analyze(MonitoredTaskValue[] mtvs)
1. Variant[] deviatedVars
2. for each mtv • mtvs
3.   do boolean isQoSOut ← isQoSDeviated(mtv)
4.   if isQoSOut then
5.     deviatedVars += getDeviatedVariant(mtv)
6. end for
7. return deviatedVars

```

In our running example we had explored the monitored task “Receive CAGED File” that has the Self-Optimization (O) as the desired autonomic principle and in its operational decomposition presents the Variation Point related to the Response Time attribute. In this sense, we explored three scenarios according the data provided in Figure 4 that are related to the tree variants presented in this Variation Point:

- Scenario 1: In the beginning of the month the reception rate is increasing time-to-time and this is an expected change in the environment according to the CAGED characteristics. The system provided the following information: ReceptionTrendIsOK=true; LastThreeCycleIncreasing=true (Contextual); and ResponseTime=382ms (QoS);

- Scenario 2: After the deadline to send the CAGED File, the competence is closed to receive files of previous month and opens to receive the next competence. As result, reception rate decreases time-to-time. Considering this, the system provided the following information: ReceptionTrendIsOK=true; LastThreeCycleDecreasing=true (Contextual); and ResponseTime=80ms (QoS);
- Scenario 3: The reception is out of the reception peak and the reception rate is increasing, as shown in Figure 4 (10/06/2011). In this case, the system must evaluate the deviation but not increase the resource. Considering this, the system provided the following information: ReceptionTrendIsOK=false; LastThreeCycleIncreasing=true (Contextual); and ResponseTime=457ms (QoS);

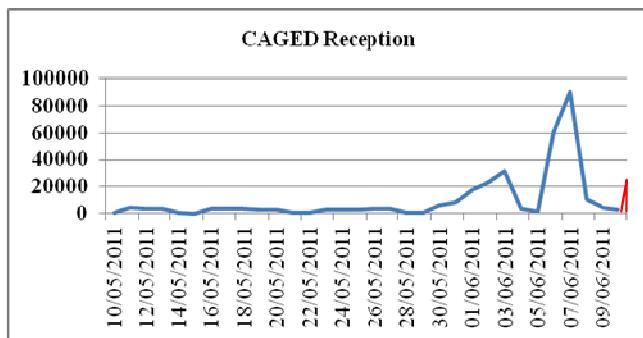


Fig. 4. CAGED Reception Rate

Considering scenario 1, 2 and 3; in different moments the analyzer module would return respectively the variants: Increase Reception Rate, Decrease Reception Rate and Deviation Reception Rate. As can be observed, it is important to emphasize that even if a quality attribute is out of the defined value, the deviated variants will only be returned only if there is contexts that enables it. For example if the system returns a response time of 350ms in the execution of the task “Receive CAGED File” and there is no context that influences in this value, the autonomic engine will not return deviations.

Plan

The main objective of planner module is planning the interventions that will be executed in the system according the information provided by the analyzer module. As shown in Figure 5, the planner module has two components: The Planner Execution Selector and the Process Status Manager. The component Planner Execution Selector reads all deviated variants obtained through analyze module and then obtain, from the Operational Level, the Operational Tasks related to these variants. The status of each selected Operational Task is managed through the Process Status Manager component to assess if this task can be executed. The status can indicate if there are other executions of the same Operational Task in the moment (in_progress) or if there are some previous plans that have the same operation and have not stated yet (uncommitted). In this case, the Operational Task does not enter in the plan that will be executed.

Regarding our running example, considering Scenario 1 (defined in section 4.2.2), the planner module returns the “Increase Resource” Operational Task. While in Scenario 2 the “Decrease Resource” Operational Task is returned. Finally, in the Scenario 3, the module returns “Analyse Deviations”.

Actuate

The actuator module has the objective of actuating in the instrumented system. For this reason, the module has two components as presented in Figure 5: The Actuator Manager and the Operational Task Assigner.

According the Operational Task returned by the planner module, the Actuator Manager accesses the Service Level to obtain the Operational Service with the parameters that it needs to be executed. The Operational Task Assigner access the instrumented system to finally perform the adaptation.

Considering our running example, if there is a plan to perform the operational task “Increase Resource”, the Actuator Manager component will access the specification of the service named “OP1”. This specification has the name of the service that is: *expandReceptionMem(ResourceType rt, ResourceValue rv, SystemModule sm)* where *ResourceType* is the type of resource that the analyst wants to increase, *ResourceValue* is the percentage of growth and the *SystemModule* is the identification of the sub-system. These values are predefined as in this case as: *rt=* “memory”; *rv=0.1*; and *sm=cag.Reception*. It corresponds to 10% of increase of memory resource.

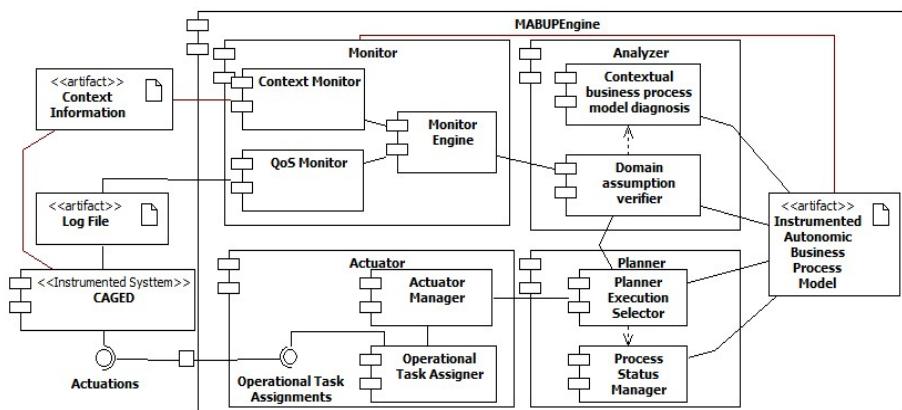


Fig. 5. MABUP Engine

5 Discussion and Conclusion

In this paper we outlined an approach for managing autonomic business processes following a multi-level strategy. An architecture of autonomic business process models that intends to represent the correct behavior of the systems from the business, technological, operational and service points of view. This proposal has been divided

into two parts: the modeling and the management phases. The modeling phase express how to obtain four abstract levels that instrument the business process model with autonomic features; namely, Organizational Level, Technological Level, Operational Level and Service Level. The management phase proposes a MAPE (Monitor-Analyze-Plan-Execute) engine that considers both instrumented ABPM (Autonomic Business Process Model) and the instrumented system obtaining information about the environment context and quality attributes to guide the adaptation.

The paper (1) defines the MABUP (Multi-level Autonomic Business Process) model used at runtime to represent the correct behavior; (2) presents the logical view on the architecture to manage the multi-level autonomic business processes; (3) introduces algorithms to perform the MAPE modules of our approach; and (4) shows how an existing system can be modeled to exploit our architecture.

The real example demonstrates the feasibility of the presented conceptual architecture and highlights its applicability to a real-life scenario. Furthermore, the benefits of our approach are manifold and include:

- *Modularity of ABPM:* We have relied on Communication Analysis principles to deal with the overhead in the monitoring phase. The Organizational Level helps to indicate mission-critical activities that must be treated in an autonomic manner. The Technological Level, a refinement of the mission-critical activities, specifies the autonomic tasks that must be monitored.
- *Scalability:* Modularity helps abstracting processes, thus reducing the complexity of ABPM, and increases the scalability of our approach;
- *Separation of concerns:* In our MABUP model the relationship between the business, technological and operational knowledge are explicitly expressed in different and interconnected abstraction levels of the model.
- *Understandability:* The different well-defined levels helps to provide understandability of ABPM;
- *Expressiveness of autonomic features in BPM:* In contrast to other approaches that need a knowledge database to express the metrics that affects the adaptation, our approach provides the concepts of critical event, monitored task, context variability and quality attributes expressed in a BPM that guide the self-management at runtime. All these concepts are interconnected to indicate how the metrics impact each autonomic business process.
- *Guide the adaptation based on metrics, as context and quality attributes, contained in the BPM:* Considering the knowledge provided in our MABUP model, the management module guides the adaptation according the context and quality attributes that affect the operational tasks.

Our approach is part of an ongoing research endeavor. As such, much remains to be done. As future work, we plan to perform a controlled experiment to empirically evaluate our proposal.

Acknowledgment. Research supported by CAPES, CNPQ and Spanish Ministry of Science and Innovation.

References

1. Espa  a, S., Gonz  lez, A., Pastor, O.: Communication Analysis: A Requirements Engineering Method for Information Systems. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 530–545. Springer, Heidelberg (2009)
2. Ganek, A.G., Corbi, T.A.: The dawning of the autonomic computing era. *IBM Systems Journal* 42(1), 5–18 (2003)
3. Gonzalez, A., et al.: Unity criteria for Business Process Modelling. In: Third International Conference on Research Challenges in Information Science, RCIS 2009, pp. 155–164 (2009)
4. Greenwood, D., Rimassa, G.: Autonomic Goal-Oriented Business Process Management. *Management*, 43 (2007)
5. Haupt, T., et al.: Autonomic execution of computational workflows. In: 2011 Federated Conference on Computer Science and Information Systems, FedCSIS, pp. 965–972 (2011)
6. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE (2003)
7. Lee, K., et al.: Workflow adaptation as an autonomic computing problem. In: Proceedings of the 2nd Workshop on Workflows in Support of Large-Scale Science, New York, NY, USA, pp. 29–34 (2007)
8. Mosincat, A., Binder, W.: Transparent Runtime Adaptability for BPEL Processes. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 241–255. Springer, Heidelberg (2008)
9. Oliveira, K., et al.: Towards Autonomic Business Process Models. In: International Conference on Software Engineering and Knowledge, SEKE 2012, San Francisco, California, USA (2012)
10. Rahman, M., et al.: A taxonomy and survey on autonomic management of applications in grid computing environments. *Concurr. Comput.: Pract. Exper.* 23(16), 1990–2019 (2011)
11. Reijers, H.A., Mendling, J.: Modularity in process models: Review and effects. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 20–35. Springer, Heidelberg (2008)
12. Rodrigues Nt, J.A., Monteiro Jr., P.C.L., de O. Sampaio, J., de Souza, J.M., Zimbr  o, G.: Autonomic Business Processes Scalable Architecture. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 78–83. Springer, Heidelberg (2008)
13. Strohmaier, M., Yu, E.: Towards autonomic workflow management systems. ACM Press (2006)
14. Terres, L.D., et al.: Selection of Business Process for Autonomic Automation. In: 2010 14th IEEE International Enterprise Distributed Object Computing Conference, pp. 237–246 (October 2010)
15. Tretola, G., Zimeo, E.: Autonomic internet-scale workflows. In: Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond, New York, NY, USA, pp. 48–56 (2010)
16. Vedam, H., Venkatasubramanian, V.: A wavelet theory-based adaptive trend analysis system for process monitoring and diagnosis. In: Proceedings of the 1997 American Control Conference, vol. 1, pp. 309–313 (June 1997)
17. Wang, Y., Mylopoulos, J.: Self-Repair through Reconfiguration: A Requirements Engineering Approach. In: 2009 IEEE/ACM International Conference on Automated Software Engineering, pp. 257–268 (November 2009)
18. Yu, T., Lin, K.: Adaptive algorithms for finding replacement services in autonomic distributed business processes. In: Proceedings Autonomous Decentralized Systems, ISADS 2005, pp. 427–434 (2005)

Multi-perspective Business Process Monitoring

Amin Jalali and Paul Johannesson

Department of Computer and Systems Sciences, Stockholm University, Sweden
`{aj,pajo}@dsv.su.se`

Abstract. Monitoring business processes is an important area in Business Process Management. This area not only supports monitoring but also enables flexibility. Thus, it has been investigated in many other areas like Business Activity Monitoring, Exception Handling, Aspect Oriented Business Process Management, etc. These areas require to define how a process instance should be monitored from different perspectives. However, current definitions are coupled to control-flow perspective, which applies some limitations. For example, we cannot define a rule to capture situations in which an account balance is read - regardless of its process.

To capture such situations, we propose an approach to define monitoring rules. This approach enables composition of rules in a way to be decoupled from a specific perspective. To validate the result, we implemented a rule editor and a monitoring service, called Observer Service. These artefacts are used to support the definition of monitoring rules and track process instances, correspondingly. Finally, we investigated the validity and relevancy of the artefacts through a banking case study.

Keywords: Business Process Management Systems, Process Monitoring, Service Oriented Architecture, flexibility.

1 Introduction

Business Process Management(BPM) is an important area that supports automation of business processes. This automation is achieved through BPM life cycle including process design, configuration, enactment and diagnosis phases [28]. This life cycle resulted in rigid business processes that are not flexible. Therefore, another phase is added, called adjustment. In this phase, the enacted process can be adjusted and executed, without repeating the whole life cycle [3,16]. Both adjustment and diagnosis phases depend on monitoring process instances, that is achieved by tracking events. Each event is a possible monitoring point in BPM [23].

Two kinds of adjustment can be performed at runtime, i.e. allowing the process instance to be deviated from process specification, or changing the process specification and migrating the process instance(s) according to new specification. These adjustments are categorized under process flexibility as 'flexibility by deviation' and 'flexibility by change' [21]. These types depend on recognition of needed points (events) in process instances, which are fulfilled through process monitoring. Therefore, more capability in capturing events results in more ability in providing process adjustment and flexibility in action.

The adjustment and flexibility should be performed for some monitoring points in a process instance. These points should be defined using some rules, called *monitoring rules*, that specify what information is a matter of interest for monitoring. Each event carries information related to different perspectives like control-flow (or process), task (or functional), operation (or application), data, resource (or organizational), time, etc [1,20].

There is an implicit order between perspectives when defining a process model. J. Becker et al, mention that “[t]he data flow is restricted by the control flow as the data flow cannot precede the control flow” [10]. These restrictions are valid in process definition. However, they limit process monitoring if we want to define the monitoring rules with the same approach.

For example, a bank manager might be interested to monitor high value financial transactions. These transactions can be occurred by different processes and activities. If we want to define monitoring rules using the control-flow and task perspectives, we should define a lot of monitoring rules to capture each task to investigate if the value of the transaction is more than the limit.

In this paper, we solve this problem by proposing an approach to define monitoring rules independent of any specific process perspective. To do that, we introduce possible monitoring points in a process instance. Then, we investigate what sort of information can be found for each point and the relation between them. As a result, we define an algorithm to evaluate monitoring rules from different perspectives. To validate our result, we developed two applications, i.e. the monitoring rule editor and the Observer Service. The editor supports the definition of monitoring rules based on investigated relation. The service monitors process instances to check if rules are satisfied or not. Moreover, we investigated the relevancy of the problem using a banking case study. The actual implementation of the study using our artifices is in progress.

Therefore, the remainder of the paper is organized as follows. In Section 2, we give a description of how the rules are defined in different BPM areas. Then, we present the relation between process perspectives and different states of monitoring life cycles in Section 3. Section 4 demonstrates the rule editor and the architecture of the implemented service. Section 5 investigates the validity and relevancy of the artefacts using a banking case study. Section 6 discusses the limitations of the work. Finally, Section 7 presents directions for future works and concludes the paper.

2 Background

A lot of areas in BPM paradigm try to define how monitoring points should be specified such as *Business Activity Monitoring(BAM)*, *Exception Handling*, *Aspect Oriented Business Process Management(AOBPM)*, etc. Therefore, different attempts have been performed to monitor these points. In this section, we look at some of these attempts in general.

Business Activity Monitoring(BAM) is defined by Gartner as a concept which enables tracking business operations and making issues visible quickly, based on

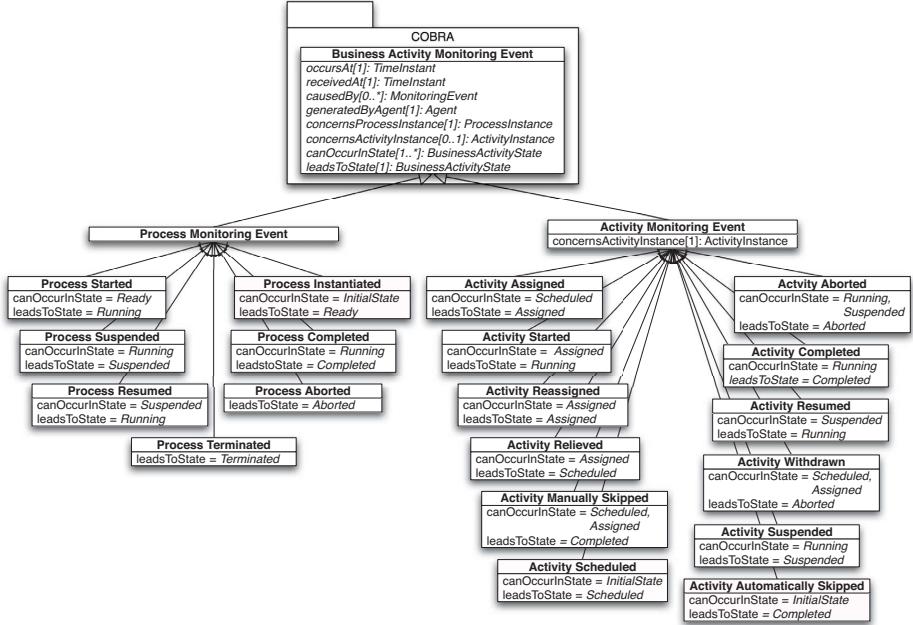


Fig. 1. Events Ontology [22]

real-time data¹. BAM could be implemented through different approaches such as process mining [27] and real-time monitoring [19]. Process mining enables tracking of business operations using event logs [7]; while, real-time monitoring detects different monitoring points in process instances and keeps track of them in action. The process mining is out of our attention, because we focus on how to define rules to capture various monitoring points in this paper. Definition of rules in BAM has been investigated in different research, e.g. [13,15,26]. For example, Pedrinaci et. al. define an event ontology for BAM in [22] (see Figure 1). The events are categorized into two groups for monitoring, i.e. process and activity. The process monitoring event consists of started, suspended, resumed, terminated, aborted, completed and instantiated events. The activity monitoring event includes assigned, started, reassigned, relieved, skipped, scheduled, aborted, completed, resumed, withdrawn, suspended and skipped. These events are points that a process instance can be monitored. However, it does not mean that we have to restrict the definition of rules to them. Such restriction can end up us with current limitations in defining monitoring rules, which are coupled to control-flow and task perspectives.

Exception Handling is an important area of BPM, which tries to support flexibility by deviation [2]. Deviations are recognized by tracking some monitoring points and evaluating some exception rules. Monitoring points are categorized into five groups, i.e. Work Item Failure, Deadline Expiry, Resource Unavailability,

¹ <http://www.gartner.com/it-glossary/bam-business-activity-monitoring>

External Trigger and Constraint Violation [8,24]. Each points can be looked from different perspectives. For example, the task that the work item is going to perform represents the functional perspective. The resource to whom the work item is allocated represents the resource perspective. Exception rules specify when an exception should be captured. Again, definition of exception rules are coupled with control-flow perspective. For example, these rules are defined using an exception event attached to an activity's boundary in Business Process Model and Notation (BPMN) [14].

Moreover, the same approach can be tracked for handling exceptions in other works like worklet [9], where the exception types could be recognized in a process instance using a set of rules, called Ripple Down Rules(RDR) [8]. These rules could be defined to monitor violations of exception types, which happens for a specific case or a workitem. This means that the constraint should be mapped to control-flow or task perspective first. This approach limits the definition of rules, which needs to be independent of these two perspectives. It means that, if we want to define a monitoring point for a resource or data perspective, we have to map it to control-flow or task perspectives. For example, if a customer wants to get notified when his or her account balance is decreased, we should find all cases and tasks which have the potential to withdraw money from the customer account. This limitation is because we are not able to define a constraint for only data perspective here.

Aspect Oriented Business Process Management(AOBPM) aims to separate cross-cutting concerns from business processes and model them separately [11,12,17]. Separated models need to be weaved at runtime. The weaving requires to check monitoring points to see whether some aspects are specified for them [18]. Monitoring points are called *join points*, and rules are called *pointcuts*. The rules are defined based on control-flow and task perspectives. This implies some limitations in defining aspects, e.g. enforcing some security policy when an extra confirmation is required for activities of a new clerk. Again, we need to define rules for all tasks of a process instance to check if a new clerk performed it or not.

To sum up, we found a general approach in definition of monitoring rules in different areas of BPM. These rules are defined based on control-flow and task perspectives. Other perspectives can be incorporated in rules when one of these two perspectives are specified. This implies a lot of limitations for defining business monitoring points for process instances. In fact, it enforces multiple definition of rules to capture a business monitoring point. Therefore, we define a new approach to capture monitoring points and defining rules. In the next section, we introduce this approach.

3 Approach

Monitoring points can be defined with the help of workitem life cycle. The life cycle is defined by N. Russell et al [25], and it is general for different Workflow Management Systems(WfMSs) [29]. It consists of the states that a workitem can have in its life. States can be changed by transitions(events) (look at the dashed

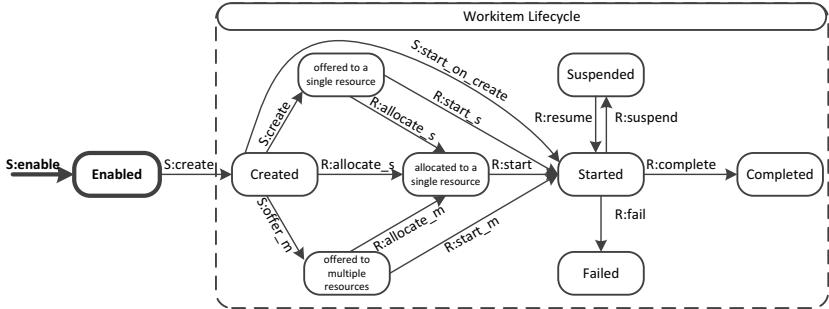


Fig. 2. Workitem Monitoring life cycle

rectangle in Figure 2). Transitions' names are started with R or S, representing whether a resource or the system(WfMS) initiates the transition. A. Rogge-Solti et al, define two more states for a workitem life cycle, i.e. *init* and *enabled* [23]- although they do not consider some other states. We could not consider these states as workitem life cycle states, since the workitem is not created yet. We also could not find any application of init state, so we exclude it. However, we consider the enabled state as one of the wokitem monitoring states because it is important from the monitoring viewpoint. Therefore, we end up with a new life cycle for monitoring, which is shown in Figure 2

The life cycle starts when the WfMS detects a workitem as enabled. All enabled workitem will not be created. For example, when a process model contains a deferred choice, many workitems can be enabled. However, as soon as one of them is created, others will not be enabled any more [5]. The created workitem can be 'offered to a single resource', 'offered to multiple resources' or 'allocated to a single resource'. Moreover, it could be started by the WfMS if it is an automated workitem. The started workitem can be suspended, and the suspended workitem can be resumed. The started workitem can also be failed or completed.

Furthermore, we should monitor the process instances, called cases. We recognized different states, which can be monitored during a case life cycle, i.e. created, completed, suspended and failed. These states could only be changed by the WfMS, so we do not incorporate the name of event initiators in event labels.

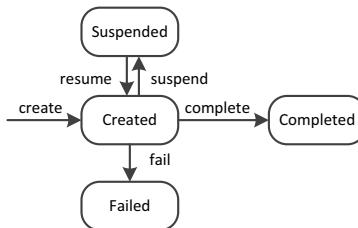


Fig. 3. Case Monitoring life cycle

Status	Control-flow	Perspectives			
		Case level	Workitem level	Task	Resource
Case Monitoring Lifecycle					
Created	+	+ (r)	-	-	-
Suspended	+	+	-	-	-
Failed	+	+	-	-	-
Completed	+	+ (w)	-	-	-
Workitem Monitoring Lifecycle					
Enabled	+	+	-	+	-
Created	+	+	+ (r)	+	-
offered to a single resource	+	+	+	+	+
offered to multiple resources	+	+	+	+	+
allocated to a single resource	+	+	+	+	+
Started	+	+	+	+	+
Suspended	+	+	+	+	+
Failed	+	+	+	+	+
Completed	+	+	+ (w)	+	+

Fig. 4. The relation between level, states and perspectives

Although there are other states during execution of a case and workitem like cancelled, there are different from one WfMS to the other. Thus, we consider the general states which exist in most of WfMSs and limit our Monitoring life cycles to existing states (see Figure 2 and Figure 3). The Workitem and Case Monitoring life cycles can be used to define monitoring points from different business process perspectives. Each process model consists of different perspectives, and each perspective exposes a different kind of monitoring points , so we should investigate what sort of monitoring points exist in each workitem monitoring state to define monitoring rules.

Figure 4 shows what monitoring points could be tracked in each state of Case and Workitem Monitoring life cycle. The control-flow monitoring points can always be captured in both life cycles. The data perspective is restricted by control-flow perspective in a way that it is always defined based on control-flow perspective [10]. Thus, we divide data perspective into two sub-categories, i.e. case level and workitem level. The case level data can be accessed in all states of both life cycles. The reading operation of data is performed when the case is created, so we added '(r)' to demonstrate this fact in Figure 4. The writing operation of data is performed when the case is completed, which is shown by '(w)' in the figure. The workitem level data can be accessed in all states of workitem life cycles, but it cannot be accessed in the enabled state of workitem monitoring life cycle.

Furthermore, The task monitoring points are available during workitem monitoring life cycle. The resource monitoring points are also available during all states of workitem monitoring life cycle except enabled and created. In these states, the resource is not yet offered or allocated, so there is no information about who will carry on the workitem.

Figure 4 indicates how the rules can be defined for monitoring process instances from different process perspective. For example, if we want to define a rule to enforce confirmation of all works that have been done by a new clerk, we should define it as intersect of workitem completed state and resource

perspective in the figure. Other cells in the same row indicate that we can limit this rule based on other perspectives. For example, we can limit this rule to enforce confirmation if the amount is greater than a limit. It is possible because we have the data perspective in this row.

To evaluate these rules, we define an algorithm which shows how different monitoring points can be analyzed. This algorithm uses basic terms, which are given as follow. The terms start with definition of perspectives, which can have any number of members. Thus, new perspectives can easily be added as a new member without changing the algorithm.

Definition 1 (Basic Definition).

- $\mathcal{P} = \{\text{Control-flow, Data, Task, Resource}\}$ is a set of Perspectives. Here, we limit ourselves to four perspectives, but they can be added just as a member of the set.
- $\mathcal{L} = \{\text{Case, Workitem}\}$ is a set of Levels. There are two levels for monitoring, i.e. Case and Workitem. Case represents the executed instance of a process model. The workitem is an executed instance of a task.
- $\mathcal{W}_{en} = \{s:\text{enable}, s:\text{create}, s:\text{start_on_create}, s:\text{offer_m}, R:\text{start_s}, R:\text{allocate_s}, R:\text{allocate_m}, R:\text{start_m}, R:\text{suspend}, R:\text{resume}, R:\text{fail}, R:\text{complete},\}$ is a set of WorkItem Event Names. These names are derived from Workitem life cycle. We also added $s:\text{enable}$ to monitor the workitem when it gets enabled.
- $\mathcal{C}_{en} = \{\text{create, suspend, resume, complete, fail}\}$ is a set of Case Event Names.
- $\mathcal{E}_n = \mathcal{C}_{en} \cup \mathcal{W}_{en}$ is a set of Event Names, which is a union of case event names and workitem event names.
- $\mathcal{E}_d = (\mathcal{P}, \text{Value})$ is EvenetData, which is a tuple. It contains a perspective and its values.
 - Value is a simple string. This string can contain, for example, xml representing the data perspective.
- $\mathcal{E}_{ds} = \{\mathcal{E}_d\}$ is a set of Event Data.
- $\mathcal{C} = \mathcal{E}_d$ is Condition. The condition is an EventData, which is a tuple containing a perspective and its values. We distinguish between event data and condition because event data is what happened in execution; while, condition is abstract representation of the situation that should be monitored.
- $\mathcal{C}_s = \{\mathcal{C}\}$ is a set of Condition.
- $\mathcal{E} = (\mathcal{E}_n, \mathcal{E}_{ds})$ is Event. The event is a tuple. It includes an event name and a set of event data. In this way, each event can carry different data from different perspectives.

Definition 2 (Monitoring Rules Definition).

- $\mathcal{M} = (\mathcal{L}, \mathcal{E}_n)$ is MonitoringPoint. It is a tuple, which contains a level and an event name. It means that a monitoring point can be any event in case or workitem level.
- $\mathcal{M}_s = \{\mathcal{M}\}$ is a set of Monitoring Points.
- $\mathcal{R} = (\mathcal{M}, \mathcal{C}_s)$ is Rule, which is a tuple. It contains a monitoring point and a set of conditions. It means that a rule defines the criteria that capture monitoring points, which can be limited from different perspectives.
- $\mathcal{R}_s = \{\mathcal{R}\}$ is a set of Rules (Ruleset).

The ruleset is used by the observer service to determine if an event satisfies conditions or not. 'Algorithm 1' shows how events can be examined to see if conditions in the ruleset are satisfied. The algorithm gets the event, level and the ruleset. It checks the rules based on specified conditions in the ruleset, and returns the set of rules, which are satisfied. The condition can have '*' as the value, which means that all values can be accepted. This algorithm is not designed for any specific perspective, so it is general. As a result, by adding any perspective to the set of perspectives the algorithm will not be changed. This algorithm are implemented in the Observer service which is described in the next Section.

Algorithm 1. Evaluate Monitoring Rules

Input: l: \mathcal{L} ,e: \mathcal{E} ,rs: \mathcal{R}_s
Output: \mathcal{R}_s

```

1:  $\mathcal{R}_s$  result;
2: for each  $\mathcal{R}$  r in rs do
3:   if  $r.\mathcal{M}=(l,e)$  then
4:     Boolean ruleResult := true;
5:     for each  $\mathcal{C}$  c in  $r.\mathcal{C}_s$  do
6:       if ruleResult=true AND c.value<>'*' then
7:         for each  $\mathcal{E}_d$  ed in e. $\mathcal{E}_{ds}$  do
8:           if ed. $\mathcal{P}=c.\mathcal{P}$  AND ed.Value<>c.Value then
9:             ruleResult := false;
10:            end if
11:          end for
12:        end if
13:      end for
14:      if ruleResult=true then
15:        result.Add(r);
16:      end if
17:    end if
18:  end for
19: return result;
```

4 Implementation

To enable definition of rules in a way that supports all combinations, we developed a rule editor and an Observer service.² In this section, we describe the rule editor and the architecture of the service.

4.1 Rule Editor

The rule editor is designed in a very generic way that can be extended easily to support other states and perspectives. It reads the perspectives and states

² Both the rule editor and the Observer Service can be downloaded from <http://people.dsv.su.se/~aj/ObserverService/>

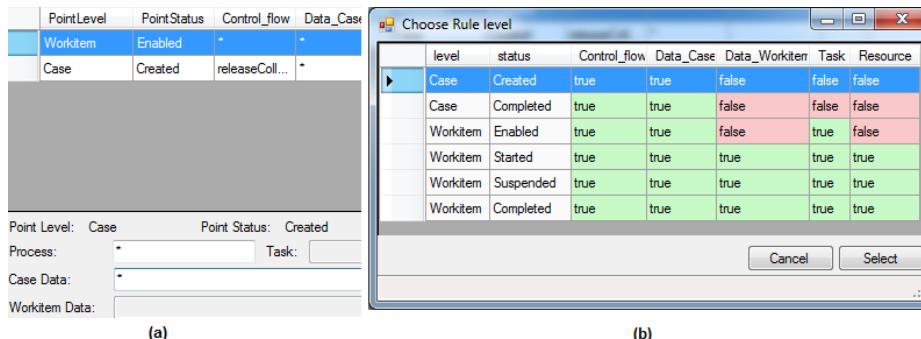


Fig. 5. RuleEditor

from an xml file. The xml indicates what information exists in each event. The result is shown to the user when s/he wants to define a monitoring rule (see Figure 5(b)). This window shows a table which is similar to Figure 4. It consists of possible levels and states for monitoring points. For each state, the editor shows what perspectives are available to be limited by monitoring rules. For example, Data_Workitem and Resource are not available for workitem enabled monitoring points. This awareness supports users to define rules, which comply to the context of events.

The user can limit the data for each perspective in the editor (see Figure 5(a)). To do that, the user should select the level (workitem or case) and the state for which s/he wants to observe the process. The editor enables the user to apply some limitation for the monitoring point based on information that exists on that point. This information can be limited from different perspectives.

For example, the bank manager might be interested to monitor all tasks that have been done by a specific clerk if s/he works on collateral which worth more than 1,000,000 USD in all processes. S/he should select the row from the table that has 'Workitem' as level and 'Completed' as the state. Then, the editor recognizes what information is available in that state, i.e. Control-flow, Data (both in case and workitem levels) and Resource. The control-flow should not be limited to any process, so '*' should be written - which indicates all processes. The Case-data should not also be limited, so '*' should be written. The Workitem-data should be limited to 1,000,000, so an xpath can be written to check the data condition, i.e. '/Collateral/Amount >1000000'. The Resource should also be limited to the specific clerk, so the name of the clerk can be written in Resource section.

This editor writes all rules in an XML file, that is used by Observer Service to monitor process instances. We limited the user to select the level and states when defining the rule. However, if the user is interested to define a rule for all levels or states, (s)he can still do that by changing the level or state field to '*' in the XML file. The architecture of the service is explained in the next section.

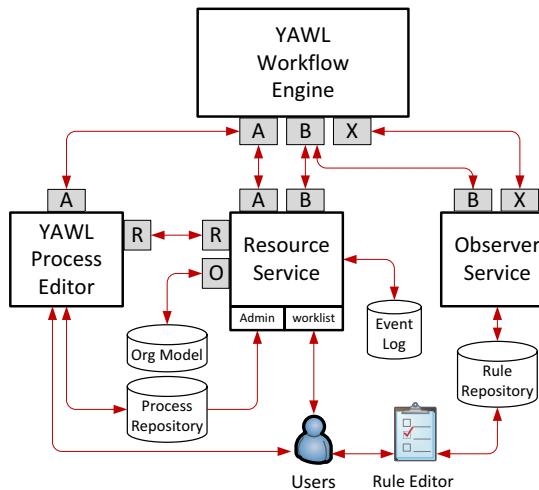


Fig. 6. Observer Services Architecture

4.2 Architecture

The Observer Service is responsible to track process instances based on monitoring rules, which are composed using the editor. The service is designed based on Service Oriented Architecture. It monitors process instances using events, which are received from WfMS. Therefore, it can be configured to observe any WfMS. We chose YAWL as a WfMS for which we monitor process instances. YAWL is selected since it supports full workitem life cycle, and it supports many workflow patterns. It also has formal definition and semantic. Moreover, it is open-source and is developed based on Service Oriented Architecture [4,6].

Figure 6 shows the architecture of our service and its relation to the WfMS. The service is connected to the YAWL Engine through two interfaces, i.e. B and X. Interface X is used to monitor case monitoring points and 's:enable' event from workitem monitoring life cycle; while, interface B is used to monitor workitem life cycle.

The resource service also plays an important role here. It is responsible to offer and allocate workitems to users. Therefore, it initiates changing some workitem states. This service collaborates with the YAWL engine through interface B to change the state of Workitems. The Interface A is utilized to upload specification to the engine, when a user launches a new process. The resource service also reads the organizational model through Interface O, which can be used for extending monitoring rules.

The Observer Service does not track other services; instead, it tracks the changes in workitem and case states through the engine. The service also reads the rules (composed by the Rule Editor) from Rule Repository. The rules specify which events should be captured. In the next section, we describe the case study which we conducted to investigate the relevancy of the artefact.

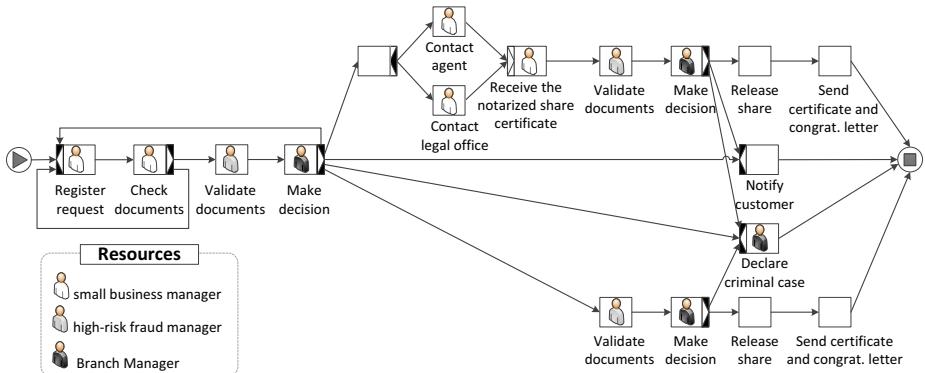


Fig. 7. Release Collateral

5 Case Study

In this section, we validate the relevance of our artefact using a banking case study. In this study, we considered different banking process, among them we chose the release collateral business process. The aim of the process is to release the collateral when the debt is not fully paid. The bank can decide about releasing the collateral based on the customer record. There are many types of collateral such as share, stock, warrant, option and co-signer. We excluded co-signer collateral since it makes the process much more complex. This complexity makes presentation of the process not possible in this paper.

Figure 7 shows this process model from the control-flow perspective. The process starts when a small business manager receives and registers documents from customer to release his or her collateral. Then (s)he checks all supporting documents. If a document is missed, then (s)he asks the customer to send complementary documents. When all documents are collected, they are sent to the high-risk fraud manager for additional review and validating the originality of documents. It is a security policy in the bank to check the originality of all documents which are received from other parties, except other banks. Then the branch manager comes up with any of the following situations:

- Declaration of criminal case: If the high-risk fraud manager detects a document as fake, the branch manager will declare the case as criminal.
- Completion of documents: The branch manager may ask for more supporting documents before any decision made to the collateral release.
- Rejection: If s/he decides the rejection, then the customer will be notified.
- Acceptance: if s/he accepts the release request, then two different activities can be performed depending on the type of the collateral:
 - If the collateral is stock, warrant or option, then the small business manager contacts the investment brokerage office and receives required information such as the most current investment activity statement. Then, the branch

manager decides on the case, i.e. rejection or acceptance. If s/he) rejects the case, the customer will get notified; otherwise, the requested collateral will be released. Then, the congratulation letter will be sent to the customer.

– If the collateral is a share, then the small business manager contacts the Investment broker and the lawyer, who had originally published the share certificate to the customer. When s/he) receives the notarized share certificate, the high-risk fraud manager should validate the originality of the document, due to security policy. If the high-risk fraud manager detects the document as fake, then the branch manager will declare the case as criminal. Otherwise, the business manager accepts the case, in most of the cases. However, if s/he) rejects the case, the customer will get notified. In case of acceptance, the share collateral will be released, and the certificate and congratulation letter will be sent to the customer.

The business manager might be interested to get notified if someone works on collateral, which has very high value. The high value is subjective and can be varied in times, so it should be determined by the business manager. Currently, we have to define monitoring points for all activities to capture such events. However, with our artefact we can get this kind of alarm by defining one rule.

The rule is defined as:

```
<rule process='releaseAsset' task='*' state='wi.completed'
      data='\\Collateral \Amount >1000000' />
```

The modelling phase of this case study is finished, and the implementation phase is still on progress. We also found out that if tasks can be categorized, it would be highly beneficial for defining monitoring points. For example, a bank manager might be interested to monitor all payment tasks, or all financial tasks. If tasks' types can be defined in process models, it can be also used when tracking them.

6 Limitations

We applied different limitations in different steps of this research such as in solution, implementation and case study.

In solution, we limit ourselves to general workitem life cycle. This means that we did not consider some states of case and workitem which are not general in WfMSs. For example, we did not consider cancelled state in both of life cycles. Moreover, we did consider limited number of perspectives to define our solution, i.e. Control-flow, Data and Resource. Therefore, our solution does not cover other perspectives like time. Although these limitations restrict our solution, it does not affect the research outcome. It is due to the fact that the solution is general and can be easily expanded to support other states and perspectives.

In implementation, we did not consider resource perspective because the YAWL engine is not responsible for that. Indeed, the resource service handles this responsibility, and the YAWL engine cannot track changes of states, which are performed by other services. If such state are going to be considered, the resource service of YAWL engine should be tracked, instead of the YAWL engine.

As a result, we had to dismiss 'offered to a single resource', 'offered to multiple resource', 'allocated to a single resource' states. This limitation does not affect the result, because other perspectives are implemented and investigated. Moreover, the implementation is general and can be easily extended to support other perspectives if the WfMS supports it.

In case study, we exclude one sort of collateral, which is used in the bank, due to reducing complexity for presentation. This limitation does not affect our goal, since we wanted to show the relevancy of the problem in the real domain. Moreover, we currently finished the modelling part of the case study, and it is not implemented completely. The implementation is in progress.

7 Conclusion and Future Works

In this paper, we presented a generic solution to monitor process instances from different business process perspectives. This solution recognizes the events as possible points that can be tracked. Each event carries different information from a different perspective. Therefore, we consider what sort of information from what perspective can be monitored in what event. The result was a set of relations that shows how the rules can be defined to comply with the process content. To validate the result, we developed two applications, i.e. a rule editor and an Observer Service. The rule editor supports definition of rules based on the defined relations. The Observer Service monitors business processes based on rules, which are defined by rule editor. In this way, we can capture events, which might be interested from a different perspective.

The relevancy of artefact is investigated by a case study in a banking domain. In this study, we chose 'releasing collateral' process. This process is used to release collateral when the customer has not paid his or her dept completely. The monitoring rule can restrict monitoring points to those in which someone works on a collateral having a high value. Despite other approaches that need to define a lot of rules for each activity to capture this business event, our artefact monitors it just by one rule. Moreover, we also distinguished the following future works:

- providing features that enable other services to subscribe for a special event.
- Using Business Rule Management System (BRMS) to define and analyse more complex monitoring rules.
- Adding more perspectives when defining rules, e.g. time, resource, cost, etc.
- Considering how our artefact can handle more exception handling in process models.
- Enabling Aspect Oriented Business Process Execution based on this artefact and investigating how much it supports separation of concerns from different perspectives.
- Considering how our artefact can extend the Business Activity Monitoring in terms of defining more measures.

Acknowledgement. We would like to appreciate Mrs. Marjan Taheri from Actax Inc. for her valuable helps in our case study. We also thank Dr. Petia Wohed to give us valuable feedback on this work.

References

1. van der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 161–183. Springer, Heidelberg (2000)
2. van der Aalst, W.M.P., Adams, M., ter Hofstede, A.H.M., Pesic, M., Schonenberg, H.: Flexibility as a service. In: Chen, L., Liu, C., Liu, Q., Deng, K. (eds.) DASFAA 2009 Workshops. LNCS, vol. 5667, pp. 319–333. Springer, Heidelberg (2009)
3. van der Aalst, W.M.P., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011 Workshops, Part I. LNBP, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
4. van der Aalst, W.M.P., Aldred, L., Dumas, M., ter Hofstede, A.H.M.: Design and implementation of the YAWL system. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 142–159. Springer, Heidelberg (2004)
5. van der Aalst, W.M.P., Barros, A.P., ter Hofstede, A.H.M., Kiepuszewski, B.: Advanced workflow patterns. In: Scheuermann, P., Etzion, O. (eds.) CoopIS 2000. LNCS, vol. 1901, pp. 18–29. Springer, Heidelberg (2000)
6. van der Aalst, W.M.P., ter Hofstede, A.H.M.: Yawl: yet another workflow language. Information Systems 30(4), 245–275 (2005)
7. van der Aalst, W.M.P., Weijters, A.: Process mining: a research agenda. Computers in Industry 53(3), 231–244 (2004)
8. Adams, M., ter Hofstede, A.H.M., van der Aalst, W.M.P., Edmond, D.: Dynamic, extensible and context-aware exception handling for workflows. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 95–112. Springer, Heidelberg (2007)
9. Adams, M., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Worklets: A service-oriented implementation of dynamic flexibility in workflows. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 291–308. Springer, Heidelberg (2006)
10. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of business process modeling. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 30–49. Springer, Heidelberg (2000)
11. Cappelli, C., Santoro, F.M., do Prado Leite, J.C.S., Batista, T., Medeiros, A.L., Romeiro, C.S.C.: Reflections on the modularity of business process models: The case for introducing the aspect-oriented paradigm. Business Process Management Journal 16(4), 662–687 (2010)
12. Charfi, A., Mezini, M.: Aspect-oriented web service composition with AO4BPEL. In: (LJ) Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 168–182. Springer, Heidelberg (2004)
13. Costello, C., Molloy, O.: Building a process performance model for business activity monitoring. In: Information Systems Development, pp. 237–248 (2009)
14. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in bpmn. Information and Software Technology 50(12), 1281–1294 (2008)

15. Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. In: Eder, J., Dustdar, S. (eds.) BPM 2006 Workshops. LNCS, vol. 4103, pp. 5–14. Springer, Heidelberg (2006)
16. Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process lifecycle. In: 10th Int'l Conf. on Enterprise Information Systems, ICEIS 2008, pp. 154–161 (June 2008)
17. Jalali, A., Wohed, P., Ouyang, C.: Aspect oriented business process modelling with precedence. In: Mendling, J., Weidlich, M. (eds.) BPMN 2012. LNBIP, vol. 125, pp. 23–37. Springer, Heidelberg (2012)
18. Jalali, A., Wohed, P., Ouyang, C.: Operational semantics of aspects in business process management. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM-WS 2012. LNCS, vol. 7567, pp. 649–653. Springer, Heidelberg (2012)
19. Kang, J.G., Han, K.H.: A business activity monitoring system supporting real-time business performance management. In: Third International Conference on Convergence and Hybrid Information Technology, ICCIT 2008, vol. 1, pp. 473–478 (November 2008)
20. Lanz, A., Weber, B., Reichert, M.: Workflow time patterns for process-aware information systems. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. LNBIP, vol. 50, pp. 94–107. Springer, Heidelberg (2010)
21. Mulyar, N.A., Schonenberg, M.H., Mans, van der Aalst, W.M.P.: Towards a Taxonomy of Process Flexibility (Extended Version). BPM Center Report BPM-07-11. BPMcenter.org (2007)
22. Pedrinaci, C., Domingue, J., Alves de Medeiros, A.K.: A core ontology for business process analysis. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 49–64. Springer, Heidelberg (2008)
23. Rogge-Solti, A., Weske, M.: Enabling probabilistic process monitoring in non-automated environments. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (eds.) EMMSAD 2012 and BPMDS 2012. LNBIP, vol. 113, pp. 226–240. Springer, Heidelberg (2012)
24. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Exception handling patterns. Process-Aware Information Systems. Technical report, BPM Center Report BPM-06-04. BPMcenter.org (2006)
25. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
26. Sadiq, S., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
27. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M.T., van der Aalst, W.M.P.: The proM framework: A new era in process mining tool support. In: Ciardo, G., Darrodeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
28. Weske, M., van der Aalst, W.M.P., Verbeek, H.M.W.: Advances in business process management. Data and Knowledge Engineering 50(1), 1–8 (2004)
29. Wohed, P., Russell, N., ter Hofstede, A.H.M., Andersson, B., van der Aalst, W.M.P.: Patterns-based evaluation of open source bpm systems: The cases of jbpm, openwfe, and enhydra shark. Information and Software Technology 51(8), 1187–1216 (2009)

Corrective Evolution of Adaptable Process Models

Luciano Baresi¹, Annapaola Marconi², Marco Pistore², and Adina Sirbu²

¹ Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy
baresi@elet.polimi.it

² Fondazione Bruno Kessler (FBK), Trento, Italy
{marconi,pistore,sirbu}@fbk.eu

Abstract. The aim of corrective evolution is to update a process model by plugging in successful process instance adaptations, while ensuring that the resulting model satisfies the goal of the original model. Corrective evolution is necessary if adapting at runtime is costly, or if adaptations fail and should be avoided. Considering that a trace is a recording of executed activities, we identify three possibilities for correcting process models, depending on the traces on which the instance adaptation should be plugged in. A correction is *strict* if the adaptation should be plugged in on a precise trace, *relaxed* if on all traces, and *relaxed with conditions* if on a subset of all traces. Automated techniques for corrective evolution are necessary since changing models manually is difficult and there is a need to verify that the evolved model satisfies the goal. We provide automated solutions for two cases: when corrections are strict, and when they are either strict or relaxed. We evaluate the tradeoffs between strict and relaxed corrections using a real log.

1 Introduction

Modeling business processes is a complex task which can be simplified by allowing process instances to be structurally adapted at runtime, based on context (e.g., by adding or deleting activities). The process model then no longer needs to include a handling procedure for every exception that can occur. Instead, it only needs to include the assumptions under which a successful execution is guaranteed. If a design-time assumption is violated, an exception is triggered and the exception handling procedure matching the context is selected or constructed at runtime (e.g., [2,9,4]). However, if runtime structural adaptation is allowed, the process model may later need to be updated based on instance adaptations. For example, model updates are necessary if dealing with a frequent exception at runtime is too costly, or if an adaptation fails and should be avoided.

Evolving the process model based on instance adaptations or process variants is not new, and has already been addressed in, e.g., [7,10,12,19]. However, an issue insufficiently addressed in the previous work is how to evolve a process model and also ensure that the evolved model continues to achieve the goal of the original model. We refer to the problem of evolving a process model based on selected instance adaptations, such that the evolved model satisfies the goal of the original model, as *corrective evolution*.

We illustrate the need for corrective evolution on a car logistics scenario, in which cars arrive from manufacturers at a sea port and must be delivered to a retailer. The car handling process includes storing the car, waiting for a delivery order, performing treatments (e.g., painting), and delivering. The goal is to deliver the car to the retailer

in perfect condition. Cars may be damaged at any point. The repair can be performed immediately or can be postponed, depending on context (e.g., availability of resources). To consider at every step that the car can be damaged, and all the contexts in which the damage can occur, would complicate the process model significantly. Instead, we specify a constraint that the car should not be damaged. If this constraint is violated, the process instance will be adapted based on context. By analyzing the logs, we determine that if the car is damaged while being stored, and other cars are waiting to be repaired, the repair should be postponed. If the car is damaged a second time in the storage area, unless many cars are waiting, the repair should not be postponed anymore, such that the car is delivered on time. We therefore need to evolve the process model, such that the repair is postponed in the first situation, and performed immediately in the second. Moreover, the evolved model must satisfy the goal to deliver the car in perfect condition.

When plugging an instance adaptation at a certain point in the process model, we need to consider that there can be multiple paths to reach this point in the model. Each path is uniquely identified by a trace, i.e., a recording of the executed activities. For each adaptation there are three options, depending on which traces the adaptation should be plugged in. To the best of our knowledge, this distinction is not present in the literature.

- The first option, which we call a *strict correction*, is to plug in the adaptation only for the trace corresponding to the instance where the adaptation was used. The advantage is that the resulting model will contain only known behavior.
- The second option, or *relaxed correction*, is to plug in the adaptation on all the traces leading to the specified point. Although more behavior is introduced, the resulting model should be smaller than the model obtained with strict corrections. The reason is that applying strict corrections requires unfolding the model.
- Relaxed corrections are not always possible, since the resulting model must also satisfy the goal. The third option, *relaxed correction with conditions*, is to plug in the adaptation for a subset of the traces leading to the specified point.

In our scenario, we obtain different process models depending on the type of the two corrections. If both corrections are strict, the second adaptation is applied only for the trace when the car is damaged a second time, after having postponed the repair for the first damage. If the second adaptation is plugged in as a relaxed correction, it will be applied also for the traces where the car is damaged for the first time.

Automating corrective evolution is necessary since changing complex process models manually is difficult, and, unless all corrections are strict, there is a need to verify that the evolved model satisfies the goal. We provide automated solutions for two cases: when all corrections are strict, and when they are either strict or relaxed. These two cases do not require searching for the traces where to plug in the adaptations. The first case also does not require verification, since both process model and adaptations are assumed to satisfy the goal, when adaptations are applied only on corresponding traces.

The contributions of this paper are:

- we extend the existing work on process evolution by considering the problem of ensuring that the evolved model continues to achieve the goal of the original model.
- we identify three ways in which instance adaptations can be plugged into the model: strict, relaxed, and relaxed with conditions, and motivate their usage.
- we provide automated solutions for two special cases of corrective evolution.

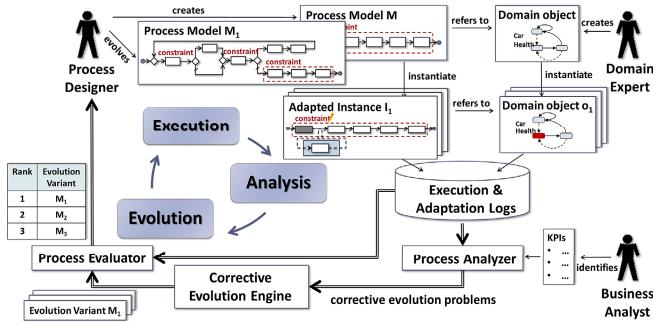


Fig. 1. Process-based application lifecycle

Section 2 presents concepts related to modeling, executing, and adapting a process-based application. We introduce the corrective evolution problem in Section 3, and solve the two cases in Section 4. We evaluate tradeoffs between strict and relaxed corrections in Section 5. Related work is discussed in Section 6, and future work in Section 7.

2 Background

We model the business logic using *process models* and the domain knowledge using *domain objects*. We relate the process to the domain by defining *goals* and process annotations based on domain objects. Fig. 1 shows the application lifecycle. In the *execution* phase, process models and domain objects are instantiated, and process instances are executed. Execution and adaptation events are recorded into logs, which are examined during *analysis*. Adaptations which are successful according to performance indicators may be selected to be included in the model, triggering *evolution*. During *evolution*, new process models are created and ranked according to, e.g., size or complexity.

2.1 Application Representation

Domain Objects. A domain object is a state transition system representing a property of an entity. States correspond to property values; value changes are transitions between states triggered by events. Controllable events are triggered by executing a process, while uncontrollable events can happen at any time and cannot be triggered directly.

Definition 1 (Domain Object). A domain object is a tuple $\langle L, L^0, \mathcal{E}, T \rangle$, where

- L is a finite set of states and $L^0 \subseteq L$ a set of initial states;
- \mathcal{E} is a set of events partitioned into sets: controllable \mathcal{E}_C , and uncontrollable \mathcal{E}_U ;
- $T \subseteq L \times \mathcal{E} \times L$ is a transition relation based on events.

Let O be a set of domain objects. We denote with P_S the set of propositions $s^S(o)$, where $o = \langle L, L^0, \mathcal{E}, T \rangle \in O$ and $s \in L$, and with $Bool(P_S)$ the set of boolean expressions over P_S . Similarly, P_E denotes the set of propositions $e^E(o)$, where $e \in \mathcal{E}$.

The domain objects in our scenario are shown in Fig. 2.

Goals. Goals specify desirable states to be reached and then continuously maintained during the execution of a process. We express goals in terms of domain objects.

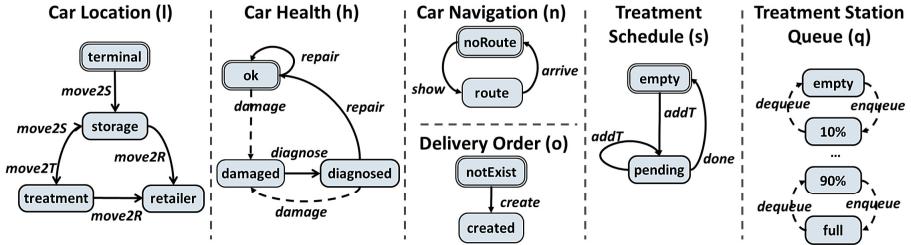


Fig. 2. Domain objects in the car logistics scenario

Definition 2 (Goal). Let O be a set of domain objects. A goal defined over O is a set of goal statements, where each goal statement is defined with the generic template $\psi_0 \implies (\psi_1 \succ \dots \succ \psi_n)$, where $\psi_i \equiv \top \mid s^s(o) \mid \psi_i \vee \psi_i \mid \psi_i \wedge \psi_i$ and $o \in O$.

A goal statement specifies that whenever the state in the left side occurs, the process should reach the state defined by the right side. If the left side is empty (\top), the state in the right side should be reached unconditionally. The states in the right side are ordered using a preference operator (\succ). In our scenario, the goal is to have the car delivered to the retailer in perfect condition: $\top \implies ok^s(h) \wedge retailer^s(l) \quad (G_1)$

Process Models. A process model is a directed graph for which the nodes are either *activity nodes* or *control connectors*, connected by *control edges*. Activity nodes are labeled with *activities*. Activities are atomic and can correspond to more than one node, i.e., duplicate nodes are allowed. We relate activities to the domain through preconditions and effects. The *preconditions* are boolean formulas over domain object states, while the *effects* correspond to controllable events. An activity can be executed only if the precondition holds, and executing the activity triggers the events in the effects.

Definition 3 (Activity). An activity is a tuple $\langle a, \text{pre}, \text{eff} \rangle$ defined over a set of domain objects O : a is the name, $\text{pre} \in \text{Bool}(P_S)$ the precondition, and $\text{eff} \subseteq P_E$ the effects.

We model the control flow using edges and the control connectors And/XorSplit, And/XorJoin. These constructs realize the patterns: sequence, parallel split, synchronization, exclusive choice, simple merge, and arbitrary loop, which form the core of any process modeling language. Our representation can therefore easily be mapped to, e.g., BPMN, WS-BPEL, or modeling languages with formal semantics.

Control edges connecting XorSplit nodes with other nodes can be annotated with *conditions*. A scope with constraint C is a sequence of activities with precondition C .

Definition 4 (Process Model). Let O be a set of domain objects and \mathcal{A} a set of activities defined over O . A process model M over O and \mathcal{A} is a tuple $\langle N, E, l, t, c \rangle$ where:

- N is a finite set of nodes partitioned into sets: N_A of activity nodes, and N_C of control connectors;
- $E \subseteq N \times N$ is a set of directed edges;
- $l : N_A \rightarrow \mathcal{A}$ is a function mapping activity nodes to activities;
- $t : N \rightarrow \{\text{Start}, \text{End}, \text{Normal}, \text{XorSplit}, \text{XorJoin}, \text{AndSplit}, \text{AndJoin}\}$ is a total function assigning a type to each node;

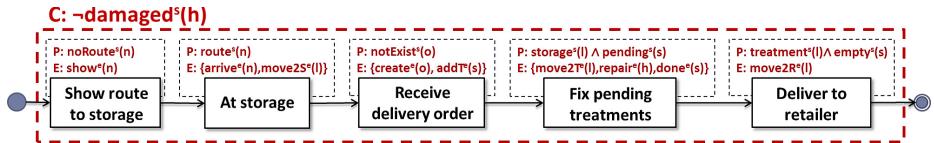


Fig. 3. Car process model

- $c : E \rightarrow \text{Bool}(P_S)$ is a branch condition function such that for all $e = (n_1, n_2) \in E$, if $c(e)$ is defined then $t(n_1) = X \text{or} \text{Split}$.

Fig. 3 shows the process model in our scenario. For example, the precondition of *Receive delivery order* is that a delivery order should not exist; the effects are that the order is created and treatments are added to the schedule. $\neg \text{damaged}^S(h)$ is constraint, which means that each activity includes this formula in its precondition.

2.2 Execution

The *trace* of a process instance is the sequence of executed activities.

Definition 5 (Trace). Let $M = \langle N, E, l, t, c \rangle$ be a process model defined over O and \mathcal{A} . A trace π on M is a sequence of activities $\langle a_1, \dots, a_k \rangle$, $k \in \mathbb{N}$, such that $\forall i, 1 \leq i \leq k, a_i \in \mathcal{A}$, and $\exists n_i \in N, l(n_i) = a_i$, with $t(n_1) = \text{Start}$. For $i < k$, n_i and n_{i+1} are such that either $(n_i, n_{i+1}) \in E$, or $\exists n'_1, \dots, n'_j \in N_C, j \geq 1$, and $(n_i, n'_1), (n'_1, n'_2), \dots, (n'_j, n_{i+1}) \in E$. Activities can occur multiple times due to loops and duplicate nodes. A trace is complete if $t(n_k) = \text{End}$, and partial otherwise.

A *configuration* is a global state of the domain objects at a certain point during the execution of a process instance.

Definition 6 (Configuration). Let O be a set of domain objects. A configuration γ of O is a total function which maps each domain object $o \in O$, $o = \langle L, L^0, \mathcal{E}, T \rangle$ to a state in L . If γ maps every object o to an initial state in L^0 then γ is an initial configuration.

Since every object state is described by a proposition in P_S , a configuration γ corresponds to an interpretation I_γ of P_S . Slightly abusing the notation, we say that γ satisfies $b \in \text{Bool}(P_S)$ and write $\gamma \models b$, if $I_\gamma \models b$. γ' is directly reachable from γ if for every $o \in O$ for which $\gamma(o) \neq \gamma'(o)$, there exists a sequence of transitions in o from $\gamma(o)$ to $\gamma'(o)$ only on uncontrollable events. Activity $\langle a, \text{pre}, \text{eff} \rangle$ is applicable in γ if there exists γ_{pre} directly reachable from γ such that $\gamma_{\text{pre}} \models \text{pre}$. Then, γ_{eff} is reachable by applying a in γ if a is applicable in γ and by applying eff to γ_{pre} we obtain γ_{eff} .

Definition 7 (Execution). Let $M = \langle N, E, l, t, c \rangle$ be a process model defined over O and \mathcal{A} . An execution of M is an alternating sequence of configurations and activities represented as $\gamma_0 \xrightarrow{a_1} \gamma_1 \dots \gamma_{k-1} \xrightarrow{a_k} \gamma_k$, where:

- $a_1, \dots, a_k \in \mathcal{A}$, and $\langle a_1, \dots, a_k \rangle$ is a trace on M ,
- $\gamma_0, \dots, \gamma_k$ are configurations of O , with γ_0 an initial configuration,
- $\forall i, 1 < i \leq k$, if $n_{i-1}, n_i \in N$ are the nodes corresponding to a_{i-1} and a_i , and $a_i = \langle \text{name}_i, \text{pre}_i, \text{eff}_i \rangle$, then:

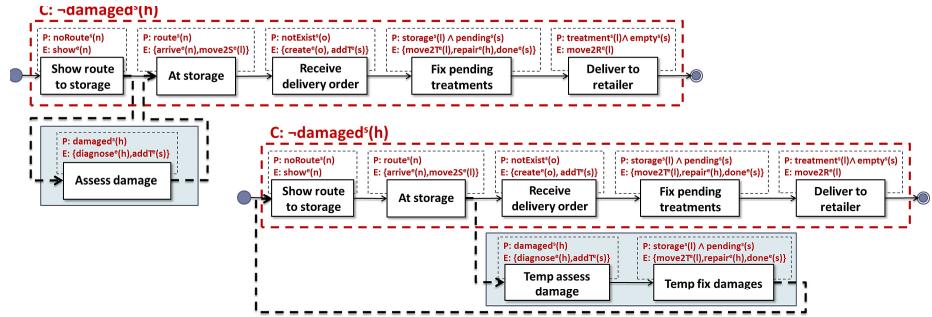


Fig. 4. Adaptation operations: (1) Schedule repair, (2) Repair temporarily

- if $(n_{i-1}, n_i) \in E$, then γ_i is reachable from γ_{i-1} by applying a_i ,
- otherwise, γ_i is reachable from γ_{i-1} by applying $a'_i = \langle \text{name}_i, \text{pre}_i \wedge \varphi, \text{eff}_i \rangle$, where φ is the conjunction of conditions between n_{i-1} and n_i .

An execution is complete if the trace $\langle a_1, \dots, a_k \rangle$ is complete, and partial otherwise.

We denote with $\text{Exec}(M)$ the set of complete executions that can be produced by model M . $\text{Exec}(M)$ can be infinite if M contains loops. M satisfies a goal statement $\psi_0 \implies (\psi_1 \succ \dots \succ \psi_n)$ if every complete execution of M is such that if a configuration satisfying ψ_0 is reached during the execution, then the last configuration satisfies at least one of ψ_1, \dots, ψ_n . M satisfies a goal G if it satisfies all the statements in G . The model in Fig. 3 satisfies goal G_1 . For every complete execution of M , every configuration reached satisfies \top , and the last configuration satisfies $\text{ok}^s(h) \wedge \text{retailer}^s(l)$.

2.3 Adaptation

Process instances can be adapted structurally while they are running: activities can be added, removed, re-executed. We start from the premise that adaptation is triggered by a constraint violation. Based on this premise, plugging an adaptation into the process model in the evolution phase results in an enhancement of the model, which allows to insert several adaptations at the same time. A second premise is that adaptation is performed to reach the goal. This ensures that there is at least one situation for which the adaptation can be plugged in the model such that the resulting model satisfies the goal. These two premises are more restrictive than that of existing process evolution approaches (e.g., [8, 10, 19]), where there are no domain-level restrictions on adaptations.

Given a process model M , an adaptation is an operation $\text{adapt}(M^a, \text{from}, \text{to})$, where M^a is a process model, and from, to are nodes in M . This operation allows to interrupt an execution of M after having completed from , execute M^a , and then resume from to . Adaptations are one-time changes, i.e., if from is reached again, M^a is not re-executed. Without loss of generality, we consider that adaptations cannot contain nodes from the main model and jumping in a parallel branch is not possible. An equivalent adaptation satisfying these conditions can be constructed by duplicating nodes.

Definition 8 (Adaptation). Let $M = \langle N, E, l, t, c \rangle$ be a process model defined over O and \mathcal{A} . An adaptation of M is an operation $\Delta = \text{adapt}(M^a, \text{from}, \text{to})$, such that:

$M^a = \langle N^a, E^a, l^a, t^a, c^a \rangle$ is a process model defined over O and \mathcal{A} , with $N \cap N^a = \emptyset$, from, to $\in N$, and to is not part of an AND-block.

An adaptation is applicable to a partial execution ω of M only if *from* is the last completed node in ω , there is at least one execution of M^a starting from the last configuration in ω , and any resulting complete execution satisfies the goal of model M .

We represent adaptation as a single operation rather than using change patterns [18], to emphasize that it is a solution to an exception. If it were represented as multiple change operations, the meaning of an adaptation as an indivisible solution would be lost. Fig. 4 shows two adaptations of the model M from Fig. 3. *Schedule repair* is applicable to M on execution $\omega_1 = \gamma_0 \xrightarrow{\text{Show route to storage}} \gamma_1$. *Repair temporarily* is applicable to M on $\omega_2 = \gamma_0 \xrightarrow{\text{Show route to storage}} \gamma_1 \xrightarrow{\text{At storage}} \gamma_2$. Let $\omega_3 = \gamma_0 \xrightarrow{\text{Show route to storage}} \gamma_1 \xrightarrow{\text{Assess damage}} \gamma_2 \xrightarrow{\text{At storage}} \gamma_3$ be an execution of M adapted by *Schedule repair*. *Repair temporarily* is applicable to M also on ω_3 , which is the case when the constraint is violated a second time.

3 Corrective Evolution

To integrate an adaptation in a process model, we need to specify the point in the model and the condition under which it must be plugged in. We specify a plug-in point as a set of traces. Given a process model M and an adaptation $\Delta = \text{adapt}(M^a, \text{from}, \text{to})$, the plug-in point must be such that each trace to this point ends with the activity corresponding to *from*. Given a plug-in point, the restrictions for the condition are:

- at least one configuration reachable at the plug-in point satisfies the condition;
- the next activities in M are not applicable in any plug-in point configuration which satisfies the condition;
- Δ is applicable to every plug-in point configuration which satisfies the condition.

Conditions which satisfy the first two restrictions are *deviating conditions*. A deviating condition represents plug-in point configurations for which the model does not specify how to proceed. A deviating condition which satisfies the third restriction is a *plug-in condition*. The combination of adaptation, plug-in point, and condition is a *correction*.

Definition 9 (Correction). *Let M be a process model defined over O and \mathcal{A} . A correction C is a tuple $\langle ct, \pi, \varphi, \Delta \rangle$ such that:*

- $ct \in \{ \text{strict, relaxed, with-conditions} \}$ is the correction type;
- π is a partial trace on M ;
- φ is a boolean expression from $\text{Bool}(P_S)$;
- $\Delta = \text{adapt}(M^a, \text{from}, \text{to})$ is an adaptation of M .

We say that C is applicable to M if:

- φ is a plug-in condition for applying Δ to M on π ,
- if $ct \neq \text{strict}$, then φ is a deviating condition for M and node *from*.

The point where Δ must be plugged in is determined by the type ct , the trace π , and the node *from*. For all correction types, Δ should be plugged in after *from*, under condition φ . If ct is strict, both φ and Δ should be applied only on π . If relaxed, they should be applied on all traces leading to *from*. Finally, if relaxed with conditions, they should be applied on at least one trace leading to *from*.

Let C be a correction applicable to M . We denote with $\text{Exec}(M, C)$ the set of complete executions that can be produced by M corrected by C . $\text{Exec}(M, C)$ adds new complete executions to $\text{Exec}(M)$. A new execution ω corresponds to a process instance for which a configuration γ satisfying φ is reached at the plug-in point. ω continues with an execution of M^a , and then resumes on M from node to . These new complete executions do not replace any complete executions of M , and $\text{Exec}(M) \subseteq \text{Exec}(M, C)$. We are interested in retrieving the process model corresponding to $\text{Exec}(M, C)$. It can be the case that a model which also satisfies the goal of M does not exist. However, a model M' such that $\text{Exec}(M') = \text{Exec}(M, C)$ can always be constructed.

Given a process model satisfying a goal, the corrective evolution problem is to apply a sequence of corrections to this model, such that the resulting model satisfies the goal.

Definition 10 (Problem). *Let M_0 be a process model and G a goal such that M_0 satisfies G . Let C_1, \dots, C_n be a sequence of corrections, such that $\forall i, 1 \leq i \leq n$:*

- $C_i = \langle ct_i, \pi_i, \varphi_i, \Delta_i \rangle$, $\Delta_i = \text{adapt}(M_i^a, \text{from}_i, \text{to}_i)$, and to_i is a node from M_0 ;
- C_i is applicable to M_{i-1} , and M_i is such that $\text{Exec}(M_i) = \text{Exec}(M_{i-1}, C_i)$.

The corrective evolution problem is to find a process model M_n which satisfies G .

Without generality loss, we assume that every adaptation returns the control to M_0 . If Δ_2 returns the control to Δ_1 , an equivalent Δ'_2 can be created by duplicating nodes.

We can now formalize the inputs to the corrective evolution problem in our scenario:

- process model M satisfying the goal G_1 from Section 2.1;
- correction $C_1 = \langle \text{strict}, \pi_1, \varphi_1, \Delta_1 \rangle$ where: $\pi_1 = \langle \text{Show route to storage} \rangle$, $\varphi_1 = \text{damaged}^S(h) \wedge (40\%^S(q) \vee \dots \vee \text{full}^S(q))$, Δ_1 is the *Schedule repair* in Section 2.3;
- correction $C_2 = \langle \text{strict}, \pi_2, \varphi_2, \Delta_2 \rangle$ where: $\pi_2 = \langle \text{Show route to storage}, \text{Assess damage}, \text{At storage} \rangle$, $\varphi_2 = \text{damaged}^S(h) \wedge (\text{empty}^S(q) \vee \dots \vee 70\%^S(q))$, and Δ_2 is the *Repair temporarily* in Section 2.3.

The solution M_n is an enhancement of M_0 , i.e., every complete execution of M_0 is also a complete execution of M_n . For every C_i , there are two possibilities. If C_i is applicable to M_0 , then M_n can replay all the executions that would result by applying C_i to M_0 . It can be the case that C_i is not applicable to M_0 , but it is applicable to M_0 corrected by C_{j_0}, \dots, C_{j_k} , $1 \leq j_0 < \dots < j_k < i$. This is the case when the adaptation must then be plugged in after other adaptations. Then, M_n can replay all the executions that result by applying C_i to M_0 corrected by C_{j_0}, \dots, C_{j_k} .

By formulating the problem as the application of n corrections, rather than only one, we are addressing a more general problem, in the sense that more solutions may be found. One reason is that we verify goal satisfaction only after applying all corrections. Moreover, by applying n corrections, the set of traces on which a relaxed correction (with conditions) can be applied changes depending on the other corrections.

4 Solution

In this section, we solve the case when all corrections are strict (*strict corrective evolution*), and the case when they are either strict or relaxed (*relaxed corrective evolution*).

In the first case, a solution can be constructed naively, by unfolding the process model up to the plug-in point, adding the adaptation, and duplicating the fragment starting

with *to*. Such a naive solution has many duplicated nodes. The challenge is to find a solution with as few duplicated nodes as possible. For this purpose, we encode the process model, domain objects, traces, conditions, and adaptations into state transition systems (STSs). We compute the parallel product of these STSs and obtain an STS which encodes all the executions of the corrected model. We use the correspondences created when encoding the inputs to translate this STS to a new process model.

In the second case, it is necessary not only to compose the process model and adaptations, but also to verify that the composition satisfies the goal. For this purpose, we devise a solution based on planning. We encode the inputs as STSs, encoding also the goal. We use the parallel product as a planning domain and create a planning goal from the process goal. We apply the approach in [1], and generate a controller for the domain to satisfy the planning goal. If a controller exists, we translate it to a new process model.

Since both solutions involve encoding the inputs as STSs, we first introduce some basic STS notions and the encoding of each input. We then present each solution.

An STS contains a set of states, some marked as initial and/or accepting. Each state is labeled with a set of properties. The STS moves to new states as a result of performing actions, which are either *input* (controllable) or *output* (not controllable).

Definition 11 (STS). *Let \mathcal{P} be a set of propositions and $Bool(\mathcal{P})$ the set of boolean expressions over \mathcal{P} . A state transition system is a tuple $\langle \mathcal{S}, \mathcal{S}^0, \mathcal{I}, \mathcal{O}, \mathcal{R}, \mathcal{S}^F, \mathcal{F} \rangle$, where:*

- \mathcal{S} is the set of states and $\mathcal{S}^0 \subseteq \mathcal{S}$ the set of initial states,
- \mathcal{I} and \mathcal{O} are the input and respectively output actions,
- $\mathcal{R} \subseteq \mathcal{S} \times Bool(\mathcal{P}) \times (\mathcal{I} \cup \mathcal{O}) \times \mathcal{S}$ is the transition relation,
- $\mathcal{S}^F \subseteq \mathcal{S}$ is the set of accepting states,
- $\mathcal{F} : \mathcal{S} \rightarrow 2^{\mathcal{P}}$ is the labeling function.

We write $s, \mathcal{F} \models b$ to denote that boolean expression b is satisfied in state s given \mathcal{F} . Transitions are guarded: (s, b, a, s') is possible in s only if $s, \mathcal{F} \models b$.

The parallel product of two STSs states that the two STSs move concurrently on common actions, and independently if there are no common actions.

4.1 Encoding into STSs

Encoding Process Models. For a process model M we construct an STS Σ_M by applying the rules in Table 1. α denotes a generic element, $s \xrightarrow{\alpha} s'$ the recursive translation of α . To differentiate input from output actions, names are prefixed with “?”, respectively “!”. For the And-block, Table 1 shows the case with two activities; the STS for a generic And-block allows every possible interleaving combination. Preconditions are copied as guards. Due to the guards, Σ_M does not move if run in isolation; the transitions become enabled in the product of Σ_M with the domain objects STSs. We label each state with a new proposition ($\mathcal{F}(s) = \{s(M)\}$), and mark initial and final states as accepting.

Encoding Adaptations. We define an STS Σ_{Δ_i} for each adaptation Δ_i . We first translate the model M_i^a . Each adaptation realizes a jump in M_0 , and, if applied on Δ_j , $j < i$, also a reset jump in Δ_j . We encode these jumps using a new action $?resume_i$ and label the start of the jump with $point_i$, to mark the point where Δ_i must be applied. The initial transitions in Σ_{Δ_i} are guarded by $\varphi_i \wedge point_i \wedge trace_i$, where $trace_i$ will mark the end of π_i . If an adaptation is started, an adaptation (not necessarily the same) has to finish for the control to be given to the main process, and the control cannot be given to

Table 1. Encoding process model elements as STSs

Process model element		STS transitions
activity node $n \in N_A$, $l(n) = \langle a, pre, eff \rangle$		$(s_b, pre, ?a, s_e)$
sequence		$s_b \xrightarrow{\alpha_1} s', s' \xrightarrow{\alpha_2} s_e$
XORSplit		$(s_b, \top, ?xor, s_0)$ $(s_0, cond, !case_1, s_1), (s_0, \neg cond, !case_2, s_2)$ $s_1 \xrightarrow{\alpha_1} s_e, s_2 \xrightarrow{\alpha_2} s'_e$
XORJoin		$s_b \xrightarrow{\alpha_1} s_e, s'_b \xrightarrow{\alpha_2} s_e$
AND-block		$(s_b, \top, ?and, s_0)$ $(s_0, \top, !order_1, s_1), (s_0, \top, !order_2, s_2)$ $s_1 \xrightarrow{\alpha_1} s'_1, s'_1 \xrightarrow{\alpha_2} s_e$ $s_2 \xrightarrow{\alpha_2} s'_2, s'_2 \xrightarrow{\alpha_1} s_e$

a previous adaptation. We encode these properties using a semaphore. Σ_{sem} has a state s_0 corresponding to M_0 , and a state s_i for every Δ_i . If Σ_{Δ_i} moves, Σ_{sem} moves from any $s_j, j < i$, to s_i . From s_i it moves to s_0 on $?resume_i$. Each state in Σ_{sem} is labeled with a flag; these flags guard the transitions in $\Sigma_{M_0}, \Sigma_{\Delta_1}, \dots, \Sigma_{\Delta_n}$.

Encoding Traces. We construct an STS Σ_{π_i} for each trace $\pi_i = \langle a_1, \dots, a_k \rangle$. Σ_{π_i} moves from state s_{j-1} to s_j on any action corresponding to a_j , and from s_{j-1} to s_{out} on actions corresponding to any other activity. Σ_{π_i} moves from s_k to s_{out} on any action corresponding to an activity. We label s_k with $trace_i$ to mark the end of π_i .

Encoding Conditions. Conditions can appear in corrections or as edge annotations. Let φ be a condition, and φ' the formula after replacing negative literals. We create an STS Σ_{φ} only if φ' contains a literal reachable through uncontrollable events. We add to Σ_{φ} two actions $!trigger_{\varphi}$ and $!no-trigger_{\varphi}$. The transitions on these actions are guarded, such that φ is triggered only before being evaluated. To simulate the uncontrollability of the events, both cases (when φ holds, and when it does not hold) will be considered.

Encoding Domain Objects. For each object $o = \langle L, L^0, \mathcal{E}, T \rangle$, we define an STS Σ_o with the same states and initial states, marking all states as accepting. We label states with the corresponding propositions, i.e., $\mathcal{F}(s) = \{s^{\mathbf{s}}(o)\}$. For every transition $(s, e, s') \in T$, we consider all the activities a for which $e^{\mathbf{e}}(o) \in eff$. Suppose a appears in M , and $?a$ is an action corresponding to a which can be executed in Σ_M from state s_j . We add to Σ_o a transition from s to s' on $?a$ guarded by $s_j(M)$ and flags. For every transition on an uncontrollable event $(s, e, s_u) \in T, e \in \mathcal{E}_U$, we consider the conditions φ containing $s_u^{\mathbf{s}}(o)$, and add a transition from s to s_u on $!trigger_{\varphi}$.

Encoding the Goal. We create an STS Σ_g for each statement $\psi_0 \Rightarrow (\psi_1 \succ \dots \succ \psi_k)$. For every ψ we introduce an action $!a_{\psi}$, guarding transitions on $!a_{\psi}$ with ψ . If $!a_{\psi_0}$ is triggered, Σ_g moves to a non-accepting state and waits for one of $!a_{\psi_1}, \dots, !a_{\psi_k}$ to be triggered, in which case it moves to an accepting state. If $!a_{\psi_0}$ is triggered again, Σ_g moves to the non-accepting state. The preference order is encoded as a requirement $\rho = (s_0, \dots, s_k)$, s_0 being the initial state, s_1, \dots, s_k the states reached with $!a_{\psi_1}, \dots, !a_{\psi_k}$.

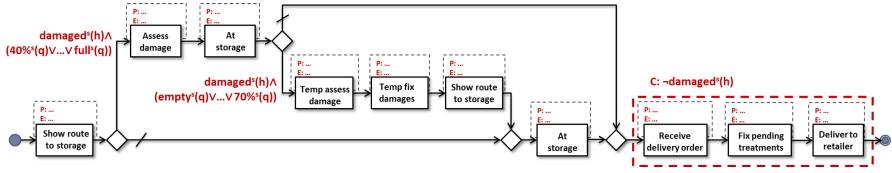


Fig. 5. Strict corrections: corrected process model

4.2 Strict Corrective Evolution

We encode the process model into an STS Σ_{M_0} , adaptations into $\Sigma_{\Delta_1}, \dots, \Sigma_{\Delta_n}, \Sigma_{sem}$, partial traces into $\Sigma_{\pi_1}, \dots, \Sigma_{\pi_n}$, conditions into $\Sigma_{\varphi_1}, \dots, \Sigma_{\varphi_m}$, and domain objects into $\Sigma_{o_1}, \dots, \Sigma_{o_p}$, as described in Section 4.1. We then compute their parallel product: $\Sigma = \Sigma_{M_0} \parallel \Sigma_{\Delta_1} \parallel \dots \parallel \Sigma_{\Delta_n} \parallel \Sigma_{sem} \parallel \Sigma_{\pi_1} \parallel \dots \parallel \Sigma_{\pi_n} \parallel \Sigma_{\varphi_1} \parallel \dots \parallel \Sigma_{\varphi_m} \parallel \Sigma_{o_1} \parallel \dots \parallel \Sigma_{o_p}$

We simplify Σ by removing the transitions which can never fire, i.e., (s, b, a, s') for which $s, \mathcal{F} \not\models b$. We can then remove the guards and the labeling function \mathcal{F} . Σ is nondeterministic, due to the fact that the domain objects STSs can have multiple initial states and nondeterministic transitions. Moreover, if an object can be in more than one state at a certain point in the process, and these states are treated in the same way, Σ will contain many similar transitions. To transform Σ back to a process model, we must first convert it to a deterministic STS. For this, as well as to remove the redundant transitions, we minimize Σ . As criteria for STS equivalence we use complete trace equivalence, one of the weakest notions of behavioral equivalence [5]. The resulting minimal, deterministic STS Σ_{strict} is transformed to a process model.

Theorem 1. Assume a corrective evolution problem defined by a process model M_0 , a goal G , and a sequence of corrections C_1, \dots, C_n , such that $\forall i, 1 \leq i \leq n, C_i = \langle ct_i, \pi_i, \varphi_i, \Delta_i \rangle, ct_i = strict$. Let M_{strict} be the translation of Σ_{strict} . Then M_{strict} is a solution for the problem defined by M_0 , G , and C_1, \dots, C_n .

The solution for the corrective evolution problem in our scenario is shown in Fig. 5. The corrected model has two new traces: one corresponding to the application of *Schedule repair*, and one to the application of both *Schedule repair* and *Repair temporarily*.

If M_0 and M_1^a, \dots, M_n^a do not contain parallelism, M_{strict} is the minimal solution to the problem, since the translation from the minimal Σ_{strict} to M_{strict} is direct. However, if any of M_0, M_1^a, \dots, M_n^a contain parallelism, this can be restored by applying post-processing techniques to M_{strict} .

With strict corrections, the original model is unfolded according to the partial traces. If an adaptation includes a backward jump, the activities in between will be duplicated in the new model. This redundancy can be removed by relaxing the corrections.

4.3 Relaxed Corrective Evolution

As in the previous case, we encode the inputs as STSs, this time encoding also the goal. If a correction C_i is relaxed, the trace is ignored and Σ_{π_i} contains one state labeled with $trace_i$. We compute the parallel product of all STSs and remove the transitions which can never fire, followed by the guards and the labeling function. The resulting STS Σ is our planning domain. We construct the planning goal ρ by combining the requirements

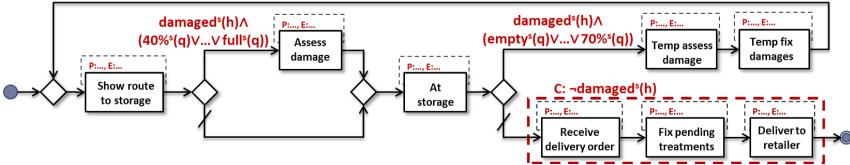


Fig. 6. Relaxed corrections: corrected process model

generated in Section 4.1. On the planning domain Σ and goal ρ we apply the technique in [1], which generates a controller Σ_c such that the controlled system satisfies the goal, i.e., $\Sigma_c \triangleright \Sigma \models \rho$. If Σ_c exists, it corresponds to a synthesis of the original model with the adaptations, which achieves the process goal. We minimize Σ_c and obtain $\Sigma_{relaxed}$, which we translate to a process model.

Theorem 2. Assume a corrective evolution problem defined by a process model M_0 , a goal G , and a sequence of corrections C_1, \dots, C_n , such that $\forall i, 1 \leq i \leq n, C_i = \langle ct_i, \pi_i, \varphi_i, \Delta_i \rangle, ct_i \in \{\text{strict, relaxed}\}$. Let $M_{relaxed}$ be the translation of $\Sigma_{relaxed}$. $M_{relaxed}$ is a solution for the problem defined by M_0 , G , and C_1, \dots, C_n .

Fig. 6 shows the model obtained in our scenario if both C_1 and C_2 are relaxed. *Schedule repair* and *Repair temporarily* are applied when the car is damaged on the way to storage, respectively at storage, independent of how many damages already occurred.

5 Evaluation

We implemented the two solutions from Section 4 into a prototype tool. For strict corrective evolution, we used the NuSMV model checker (nusmv.fbk.eu). For relaxed corrective evolution, we used WSYNTH, a tool from the ASTRO toolset (astropoint.org).

We evaluated our approach using the event log from the 2012 BPI Challenge. This is a real-life log taken from a financial institute, containing 262.200 events in 13.087 traces. The traces correspond to a loan application process. As a first step, we obtained a rough process model by filtering the log to include the most frequent traces and events, and mining the filtered log. We designed our domain objects based on the log and descriptions, and used these objects to define our goal and annotate the activities appearing in the log. We computed the differences between the model and the traces in the log, and used the most frequent differences to generate strict and relaxed corrections. We then evaluated the tradeoffs between strict and relaxed corrections along three dimensions:

- *fitness* - how much of the behavior in the log is captured by the corrected models;
- *precision* - how much extra behavior is introduced in the corrected models;
- *structure* - how much the corrected models deviate structurally from the original.

To realize these comparisons, we used several metrics devised for evaluating process mining results, which are implemented in the ProM framework (www.promtools.org).

To measure fitness, we used the f metric from [14], implemented in ProM as *token-based fitness*. This is a fine-grained metric quantifying the extent to which the traces in the log can be replayed on the process model. We were interested not only to compare the two correction types, but also to evaluate the fitness of corrected models over time.

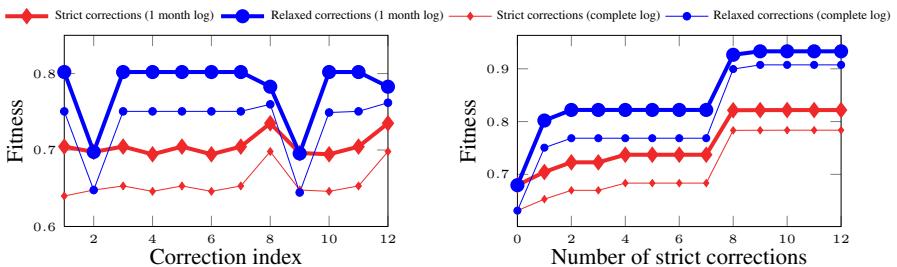


Fig. 7. Fitness: corrections applied (1) individually; (2) incrementally

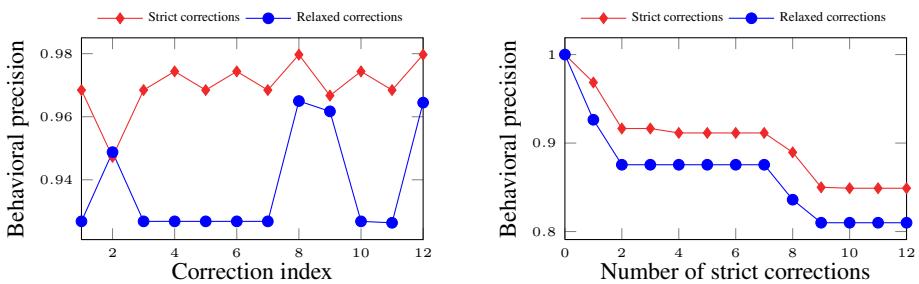


Fig. 8. Behavioral precision: corrections applied (1) individually; (2) incrementally

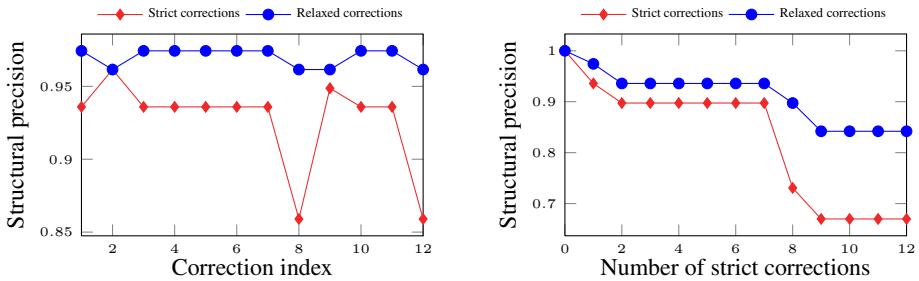


Fig. 9. Structural precision: corrections applied (1) individually; (2) incrementally

To simulate the passing of time, we used only a fragment of the entire log (roughly one sixth, corresponding to the first month) to generate our corrections. We measured the fitness of corrected models on the log fragment as well as on the entire log.

Fig. 7(1) shows the fitness of corrected models when corrections were applied individually, i.e., at each step we applied exactly one correction on the original model. Fig. 7(2) shows the fitness when corrections were applied incrementally. At step 0, we measured the fitness of the original model. Then, in the strict version, at step n we corrected the original model with n strict corrections. In the relaxed version, at step n we corrected the original model with $m \leq n$ relaxed corrections. The number of strict and relaxed corrections is not necessarily equal, since several strict corrections may correspond to the same relaxed correction. As can be seen in Fig. 7, the fitness increases

when corrections are applied, for both correction types. However, the fitness is higher for relaxed corrections, and remains higher also when tested against the entire log.

To measure changes in behavior, we used the *behavioral precision* B_P metric [11]. B_P quantifies how much extra behavior a process model allows with respect to a reference model and a log. B_P is lower when the deviation in behavior is higher. As for fitness, we measured B_P of corrected models when corrections are applied individually and incrementally (Fig. 8). The log used was the one-month log fragment. As expected, we observe that strict corrections introduce less behavior than relaxed corrections.

To measure the deviation in structure, we used the *structural precision* S_P metric [11], which assesses how many connections a process model has that are not in a reference model. Like B_P , S_P is lower when the deviation is higher. We measured S_P of corrected models when corrections are applied individually and incrementally (Fig. 9). We observe that relaxed corrections lead to smaller structural changes. An interesting case is that of corrections 8 and 12, which lead to the lowest values for the strict corrections in Fig. 9(1). For these two strict corrections, the trace passes through an And-block, and to apply the correction the model had to be unfolded up to the plug-in point. No such unfolding was necessary for the corresponding relaxed corrections.

We conclude that for this scenario there is a tradeoff between strict and relaxed corrections: relaxed corrections introduce more behavior, but lead to a higher fitness and less structural changes than strict corrections. In general, we expect relaxed corrections to introduce more behavior and less structural changes as soon as there is more than one trace to the plug-in point. Regarding fitness, a relaxed correction should be more effective if the adaptation is commonly applied at the given point, independent of trace.

6 Related Work

We focus on approaches which use the structural adaptations of process instances to support process evolution. If execution and adaptation logs are available, they can be analyzed to facilitate change reuse (e.g., [16,15]), support process diagnosis (e.g., [6]), and evolve the process model (e.g., [3,19]). Our approach belongs to the last category. In [16], process instances are grouped based on contextual properties, paths, and outcomes, to provide recommendations for improving the process model. For declarative processes, execution recommendations are generated in [15] based on past executions and optimization goals. In [6], process mining techniques are applied to change logs to provide an overview of when and why change was necessary. Also using process mining, [3] repairs a process model with respect to a log, such that the repaired model can replay the log and is similar to the original model. In [19], case-based reasoning is used to log instance changes, and derive suggestions for process model changes.

Alternatively, the result of structural adaptation can be represented as variants of a process model. Techniques for managing variants which use a single reference model to represent a set of variants (e.g., [7,8,10,13]) can also be used for process evolution. In [7] and [13], the reference model incorporates variation points, to distinguish the parts common to all variants from the variant-specific parts. The technique in [8] is used for resolving differences between variants. In [10], a heuristic search is employed to find a process model such that the distance between the model and variants is minimal.

The approaches in [19] and [10] are closest to our work, since they generate new process models based on an adaptation log, respectively process variants. Both approaches derive model changes from frequent instance adaptations. The context of the adaptations is considered in [19], but not in [10]. Further, the trace for which adaptations should be applied is considered implicitly in [10], and not considered in [19]. However, an adaptation is tightly coupled to the context *and* trace for which it is used, and may even be harmful if used for different contexts or traces. When evolving the model based on adaptations, the contexts and traces must be considered as well. If traces are ignored, we need to consider the overall goal of the process, to make sure that the adaptations introduced in the model are not harmful. The goal is not considered in [19], nor in [10]. The relation between context, traces, and goals has been considered in [16]. However, in [16] the aim is to recommend improvements to the process model, rather than to actually change it, and can be used as an analysis technique preceding corrective evolution.

Although goal compliance is insufficiently investigated for process evolution, there are many approaches which address the goal compliance of process models and their runtime adaptation, e.g., [2,9,4]. Also related is the problem of service composition, where a composite service is generated from service interfaces and goal specifications. Among service composition approaches, ASTRO [1] is particularly relevant, since we have used its powerful planning techniques to implement relaxed corrective evolution.

Finally, another relevant area is that of process model refactoring. Of the 11 techniques in [17], our work can be used for implementing RF11, *Pull Up Instance Change*.

7 Conclusions and Future Work

We presented a new approach for evolving process models based on instance adaptations. Our approach ensures that the evolved model achieves the goal of the original model. We identified three different ways for plugging adaptations into the model, and designed automated solutions for two special cases. Finally, we evaluated the tradeoffs between strict and relaxed corrections on a scenario built on a real log. In the future, we will design and evaluate solutions for the general case, when corrections can also be relaxed with conditions. The traces on which such a correction should be applied must be determined through search. The problem gets significantly more complex if more than one such correction should be applied, due to the combinatorial explosion. To deal with this complexity, we will devise heuristic techniques.

References

1. Bertoli, P., Kazhamiakin, R., Paolucci, M., Pistore, M., Raik, H., Wagner, M.: Control Flow Requirements for Automated Service Composition. In: Proc. ICWS 2009, pp. 17–24 (2009)
2. Bucciarone, A., Marconi, A., Pistore, M., Raik, H.: Dynamic adaptation of fragment-based and context-aware business processes. In: Proc. ICWS 2012, pp. 33–41 (2012)
3. Fahland, D., van der Aalst, W.M.P.: Repairing process models to reflect reality. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 229–245. Springer, Heidelberg (2012)

4. Friedrich, G., Fugini, M., Mussi, E., Pernici, B., Tagni, G.: Exception handling for repair in service-based processes. *IEEE Trans. Software Eng.* 36(2), 198–215 (2010)
5. van Glabbeek, R.J.: The linear time-branching time spectrum (extended abstract). In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 278–297. Springer, Heidelberg (1990)
6. Guenther, C.W., Rinderle-Ma, S., Reichert, M., van der Aalst, W.M., Recker, J.: Using process mining to learn from process changes in evolutionary systems. *Int'l J. of Business Process Integration and Management* 3(1), 61–78 (2007)
7. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the provop approach. *Journal of Software Maintenance* 22(6-7), 519–546 (2010)
8. Küster, J.M., Gerth, C., Förster, A., Engels, G.: Detecting and resolving process model differences in the absence of a change log. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 244–260. Springer, Heidelberg (2008)
9. de Leoni, M., Mecella, M., De Giacomo, G.: Highly dynamic adaptation in process management systems through execution monitoring. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 182–197. Springer, Heidelberg (2007)
10. Li, C., Reichert, M., Wombacher, A.: Discovering reference models by mining process variants using a heuristic approach. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 344–362. Springer, Heidelberg (2009)
11. de Medeiros, A.K.A., van der Aalst, W.M.P., Weijters, A.J.M.M.: Quantifying process equivalence based on observed behavior. *Data Knowl. Eng.* 64(1), 55–74 (2008)
12. Ploesser, K., Peleg, M., Soffer, P., Rosemann, M., Recker, J.: Learning from context to improve business processes. *BPTrends* 6(1), 1–7 (2009)
13. Reijers, H.A., Mans, R.S., van der Toorn, R.A.: Improved model management with aggregated business process models. *Data Knowl. Eng.* 68(2), 221–243 (2009)
14. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. *Inf. Syst.* 33(1), 64–95 (2008)
15. Schonenberg, H., Weber, B., van Dongen, B.F., van der Aalst, W.M.P.: Supporting flexible processes through recommendations based on history. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
16. Soffer, P., Ghattas, J., Peleg, M.: A goal-based approach for learning in business processes. In: Intentional Perspectives on Information Systems Eng., pp. 239–256. Springer (2010)
17. Weber, B., Reichert, M.: Refactoring process models in large process repositories. In: Belahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 124–139. Springer, Heidelberg (2008)
18. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowl. Eng.* 66(3), 438–466 (2008)
19. Weber, B., Reichert, M., Rinderle-Ma, S., Wild, W.: Providing integrated life cycle support in process-aware information systems. *Int. J. Cooperative Inf. Syst.* 18(1), 115–165 (2009)

Demonstrating the Effectiveness of Process Improvement Patterns

Matthias Lohrmann and Manfred Reichert

Ulm University,
Databases and Information Systems Institute
`{matthias.lohrmann,manfred.reichert}@uni-ulm.de`

Abstract. Improving the operational efficiency of processes is an important goal of business process management (BPM). There exist many proposals with regard to process improvement patterns (PIPs) as practices that aim at supporting this task. Nevertheless, there is still a gap with respect to validating PIPs in terms of their actual business value for a specific organization. Based on empirical research and experience from consulting projects, this paper proposes a method to tackle this challenge. Our approach towards *a-priori validation of process improvement patterns* considers real-world constraints such as the role of senior stakeholders and opportunities such as process mining techniques. In the sense of an experience report, our approach as well as results are illustrated on the basis of a real-world business process from human resources management, covering a transactional volume of about 29,000 process instances over the period of one year. Overall, our proposal enables practitioners and researchers to subject PIPs to a sound validation procedure before taking any process implementation decision.

Keywords: Business Process Governance, Business Process Design, Business Process Optimization, Process Mining, Process Intelligence.

1 Introduction

Research on business process management (BPM) and process-aware information systems (PAIS) has resulted in a multitude of contributions discussing options to improve the quality, performance, and economic viability of business processes. Examples range from individual “best practices” [1] to comprehensive business process quality frameworks [2]. In this context, we refer to *process improvement patterns (PIPs)* as abstract concepts to enhance particular aspects of processes. As an example, consider the “knock-out” re-arrangement of tasks to reach a decision within an appraisal process as early as possible [3].

To realize the full potential of PIPs in terms of practical relevance, it is necessary to demonstrate their actual *business value* to practitioners, thus enabling corresponding implementation decisions. To address this challenge, there exist many propositions to empirically establish the effectiveness of PIPs, including rather anecdotal evidence [4] as well as case studies [5] and survey-based research

[6]. Commonly, these approaches are based on an *ex-post* appraisal of qualitative evidence given by process managers or other stakeholders to obtain general insights applicable to analogous cases.

However, there is still a notable gap with regard to *a-priori* validation of PIPs considering a *particular application context*, which ranges from an organization's strategy and goals to its existing business process and information systems landscape. In particular, this gap should be addressed for three reasons:

1. *Similar to design patterns* in software engineering [7], PIPs constitute abstract concepts which may or may not be useful in a particular context. Experience from other scenarios is thus not sufficient to take sound organizational or PAIS-related implementation decisions.
2. *Ex-post evidence* is usually obtained from persons involved in the corresponding implementation projects, which leads to a source of bias. In addition, *a-priori* evaluation allows addressing a far wider spectrum of PIPs since it is not necessary to conclude implementation projects before a PIP can be assessed.
3. *Contemporary technology*, combining PAISs with process intelligence and process mining tools [8,9,10], provides novel opportunities to quantitatively and qualitatively gauge actual business processes, which should be leveraged for scenario-specific PIP validation.

Reflecting these considerations, this paper contributes an approach towards *a-priori* evaluation of PIPs for specific application scenarios on the basis of today's technical means. This enables sound implementation decisions, and extends the relevance of corresponding propositions from BPM and information systems (IS) research. Our approach considers both scientific rigor and practical requirements, and is demonstrated through an experience report covering a substantial real-world business process. In particular when discussing the validity of our research design, execution and results with practitioners, we made a number of general observations on what will and what won't work in *a-priori* empirical PIP evaluation. Since these may be helpful as lessons learned for practical application as well as future research, they are highlighted as *key principles*.

The remainder of this paper is structured as follows: Sect. 2 describes the sample process we use to illustrate our approach and results. Sect. 3 illustrates our approach towards addressing the research issue. Sect. 4 describes the actual evaluation of PIPs and sample results. Sect. 5 and Sect. 6 discuss related work and conclude the paper.

2 Sample Case: Applications Management Process

The business process we use to illustrate the concepts presented in this paper stems from the field of human resources management. It addresses the handling of incoming job applications as implemented in a professional services firm.

On the basis of discussions with stakeholders and the results of process mining, we can model the sample business process as presented in Fig. 1, using Business

Process Model and Notation (BPMN) [11]. For the sake of brevity, we slightly simplify the model. If an application is received, the organization has to decide whether to offer a job contract to the applicant. To this end, documents are checked first by the recruiting function, and then by the business unit. If both checks are passed, an interview is planned and executed. If the business unit confirms its interest in hiring the candidate after the interview, approval by senior management is obtained before a job contract can finally be offered to the candidate who may then accept or decline the job offer.

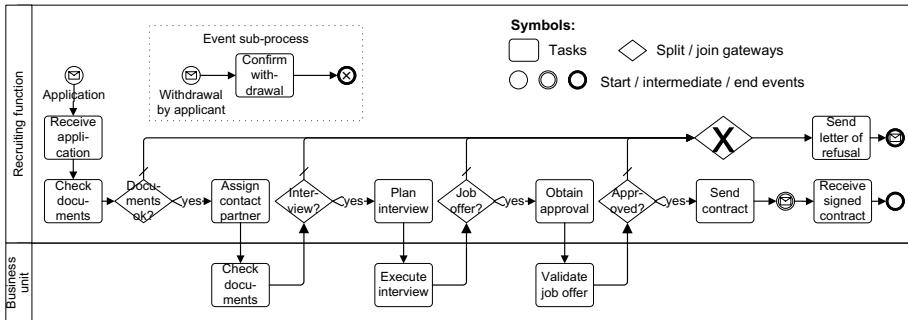


Fig. 1. Sample Process: Handling Incoming Applications (BPMN Notation)

Fig. 2 provides an overview on the distribution of termination states for our sample process covering a time period of one fiscal year, i.e., the frequency of possible states an application process instance might terminate in. We will refer to this overview when discussing our research execution in Sect. 4. A corresponding data sample obtained from the log database tables of the corresponding PAIS (in this case, a SAP ERP system) was used for our analyses. The data sample includes 27,205 cases (i.e., process instances) traceable in the PAIS. The 1,972 cases not included comprise, for example, cases handled in the business units without involvement of the human resources (HR) department.

Additional facts and empirical analyses generated in the process mining tool Disco and the statistical tool Minitab for the data sample are available in [12]. Note that Disco and Minitab were selected as examples of tools available to practitioners, alternatives might be employed as well.

3 Methodology

Like other IS artifacts, PIPs constitute *goal-bound artificial constructs* in the sense of the design science paradigm [13] to be evaluated in terms of “value or utility” [14]. In our context, this results in a particular challenge: while PIPs constitute abstract concepts applicable to a broad range of scenarios, their *business value* must be determined with regard to a particular case to enable actual implementation decisions. To this end, we employ an extended conceptual framework summarized in Fig. 3.

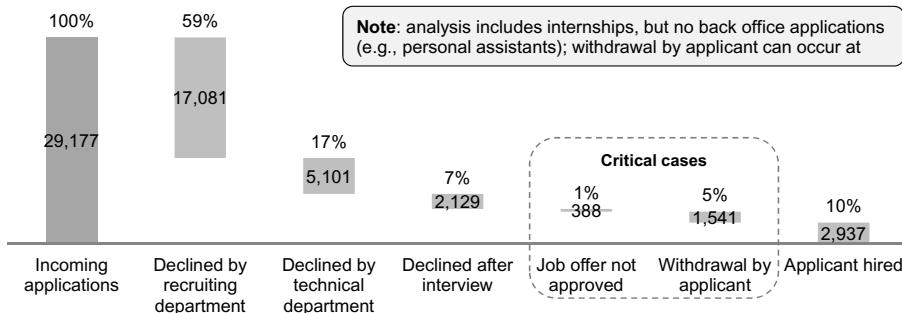


Fig. 2. Termination States of the Application Process: One Fiscal Year Sample

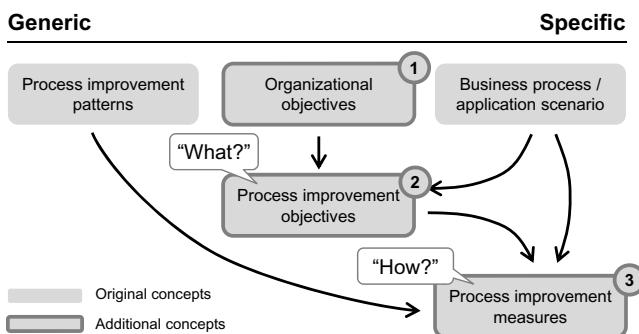


Fig. 3. Extended Conceptual Framework

Beyond the concepts of PIPs and business processes or application scenarios, we introduce organizational objectives, process improvement objectives, and process improvement measures:

1. **Organizational objectives** reflect strategic goals an organization wants to achieve, like, for example, cost savings. In principle, respective objectives are generic, but how they are prioritized against each other is specific to an organization's strategy.
2. **Process improvement objectives (PIOs)** comprise characteristics that enhance a process considering organizational objectives. PIOS may be evident, like lowering cost, or they may require additional validation. Consider, for instance, short cycle times: while it is not necessarily a strategic goal to enact processes as fast as possible, this may be a PIO if a link between cycle times and cost can be demonstrated. PIOS provide an additional layer of abstraction as a “shortcut” between improvement measures and organizational objectives. For the above example, measures might be validated by demonstrating a positive impact on cycle times instead of

overall cost. Note that the concept of PIOs corresponds to the requirement to identify stakeholders' goals as demanded in [15].

3. ***Process improvement measures (PIMs) are bundles of actions considered jointly for implementation.***¹ They reflect the application of PIPs to a specific process to realize PIOs. Note that, in many practical cases, multiple PIPs are bundled into one PIM to be jointly implemented, depending on the application context. Accordingly, a PIM is a set of one or more PIPs associated with a specific business process to address one or more particular PIOs. A-priori validation of PIPs for a particular application context thus amounts to assessing the business value of the corresponding PIMs considering relevant PIOs.

Note that, following the arrows, Fig. 3 can also be read as a methodological *top-down approach* for process improvement that has proven its value in practice, and is applied in Sect. 4: General organizational objectives are refined to PIOs specific to the business process or application scenario. Then, corresponding PIPs are selected from a generic set to be bundled into concise PIMs, again under consideration of specifics of the business process or application scenario.

Key Principle 1 (Top-down Process Improvement Methodology).

We found top-down methodologies aiming at process improvement to be the most stable in terms of change management. In principle, this paradigm is based on an early senior management agreement on the general "call for action" (see the concept of organizational objectives), which is then refined into process-related objectives (see the PIOs concept), and finally into individual improvement measures considering the specifics of the process in question. PIPs or industry-specific "best practices" (e.g., [16]) can be used to identify measures. The advantage of this top-down approach lies in focusing the discussion of individual measures on *how* things are to be achieved instead of *what* to achieve in general, and in facilitating change management through an early agreement on basic principles.

In the sense of this paper, business processes and PIMs as our unit of study are implemented by means of PAISs. To maintain scientific rigor, their assessment should take into account requirements posed towards the empirical evaluation of propositions in software engineering or IS research. In [15], Wieringa et al. subsume methodological considerations on scientific evaluation in IS engineering. Accordingly, this section proceeds by aligning the basic requirements described in [15] with regard to *problem statement* and *research design* to the conceptual framework of Fig. 3.

¹ In this context, the term "measure" is *not* to be understood as an means of measuring something (e.g., a performance indicator) or as an unit of quantity.

3.1 Problem Statement

Our problem statement (cf. Fig. 4) is structured along the requirements towards effective empirical research [15]. It refines general principles for empirical validation of PIPs as appropriate for our sample case, and describes relevant key success factors. Note that the research question refers to PIMs instead of PIPs. This characteristic reflects our goal of *scenario-specific* validation.

Components of the problem statement	Research Question "What do we want to know?"	Unit of Study "About what?"	Relevant Concepts "What do we know in advance?"	Research Goal "Why do we want to know?"
Application to empirical validation of PIPs	> Should PIMs be implemented to improve on organizational objectives?	> The business process in question > Proposed PIMs comprising PIPs	> Related work: conceptual work on the PIPs, case studies for other processes etc.	> Implementing PIMs will result in cost and risks incurred (e.g., process disruptions). Thus, implementation decisions should be based on sound investigation
Sample Case-specific refinement	> Should PIMs be implemented to reduce cost per hire (cf. Section 4.1)?	> Application management process (cf. Section 2) > Selected PIMs (cf. Section 4.3)	> Cost per hire benchmark (cf. Section 2) > Available research on "knock-out" processes	
Key success factors	> Proper demonstration of the relevance of organizational objectives to be addressed	> Proper selection of relevant PIPs to be bundled into scenario-specific PIMs	> Proper research into available literature > Use of available organizational knowledge	> Proper consideration of related investments, transactional volume etc.

Fig. 4. Problem Statement: Structure and Application

3.2 Research Design

Proper research design is the second requirement towards any scientific evaluation [15]. Fig. 5 summarizes our considerations in this regard.

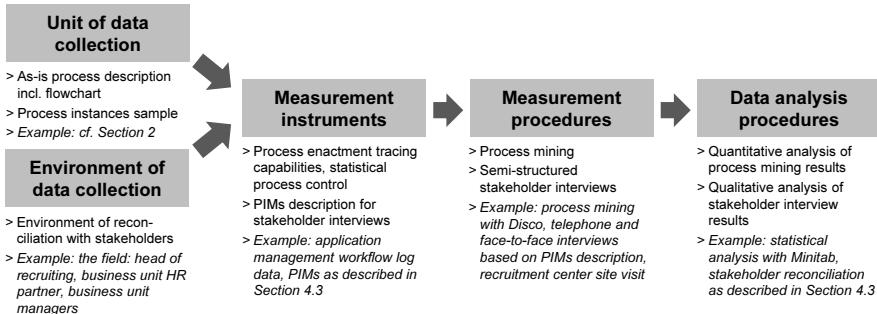


Fig. 5. Research Design

As discussed, in practical scenarios, PIPs are bundled into PIMs addressing PIOs. Accordingly, the scenario-specific business value of PIPs can be demonstrated by validating the respective PIMs' positive impact on PIOs which are, in turn, desirable with respect to organizational objectives. Our research design seeks to establish these characteristics by combining quantitative analysis based

on, for example, process enactment logs, and qualitative discussion with stakeholders. Selected methods must be properly aligned to the application context:

Key Principle 2 (Appropriate Qualitative or Quantitative Demonstration of Business Value). For each PIO to be addressed by PIMs, the underlying business value must be empirically demonstrated based on proper qualitative or quantitative analyses with respect to organizational objectives. Likewise, the business value of PIMs must be made transparent through appropriate analyses. While applicable methods are summarized in Fig. 5, the specific approach for individual PIOs and PIMs must consider the actual application context in order to balance expected insights against analysis effort. For example, the omission of obviously redundant tasks not contributing to the business objective of the process can be justified by a short qualitative description. In contrast, the introduction of additional control tasks to diminish defects later on in the process will require careful quantitative weighing of pros and cons.

Note that, regardless of the appropriate method of demonstrating business value, all types of PIOs and PIMs require the context of a concise description and a well-understood business process, and the cooperation of knowledgeable stakeholders in the field as unit and environment of data collection (cf. Fig. 5).

Key Principle 3 (Identifying Relevant Stakeholders as Interview Partners). To ensure the validity of measurement procedures, proper selection of interview partners is of particular relevance for PIOs and PIMs that should be validated qualitatively. For BPM scenarios, it is important to interview experienced senior personnel overlooking the end-to-end business process, and to represent both the “supplier” and the “customer” perspective to avoid lopsided optimization. For our sample process, we interviewed the head of recruiting operations and the administrator of the application management workflow from the “supplier” side, and the HR partner of a business unit as well as team managers from the business unit from the “customer” side.

4 Sample Case: Process Improvement Patterns Validation

We now apply the conceptual framework from Fig. 3 and our research design from Sect. 3.2 to the process described in Sect. 2.

4.1 Organizational Objectives

As discussed in Sect. 3.1, obtaining clarity about the content and business value of organizational objectives is an important prerequisite to ensure relevance of PIP validation. For our process, the following considerations apply:

Reducing Cost per Hire as Organizational Objective. In our sample application field (i.e., recruiting), organizations aim at filling vacant positions quickly, cost-effectively, and with suitable candidates. To achieve this goal, personnel marketing is tasked to generate a sufficient number of suitable candidates while the purpose of our sample process, application management, is to convert applications into actual hires. In this context, the *cost per hire* key performance indicator captures the total cost of both personnel marketing and applications management. While recruiting cost spent per application is proprietary data, based on client projects experience we may assume an amount of about 400 Euros. In our sample scenario, generating and managing 29,000 applications per year would thus sum up to 11.6m Euros total cost. Accordingly, cost per hire may be assumed to be around 4,000 Euros. Since hiring cost for talent in professional services will be higher than in, e.g., manufacturing, this value corresponds well to the average of 4,285 USD reported as cost per hire for larger organizations by a benchmarking organization [17], and seems rather conservative considering that professional recruiting consultants commonly charge half a year's salary for successful hires, depending on industry. This calculation demonstrates the relevance of reducing cost per hire through an improved application handling process.

4.2 Process Improvement Objectives (PIOs)

PIOs pertain to characteristics of the business process that affect the organizational objectives we want to improve on. Thus, they serve as a “shortcut” to facilitate the discussion of the business value of PIMs without reverting to fundamental objectives. In our sample case, cost per hire is driven by the general efficiency of the application management process, but also by its effectiveness with regard to avoiding the termination states marked as “critical” in Fig. 2:

- *Not approving a job offer* after a successful interview may be caused by defective steering of capacities (i.e., job vacancies), defective communication of terms to be offered, or defective review of application documents.
- *Job offers declined by applicants* means that the applicant does not approve of conditions offered, did not have a good impression during the application process, or has decided to take another job offer.

Since terminating the process in these states means that significant effort has been incurred with no business value in return, organizational objectives are clearly violated: On average, only one out of six applications will successfully pass interviews. However, considering defective termination events (cf. Fig. 2), only one out of ten applicants can be hired. In other words, if the process enactment defects lined out could be fully eliminated, only about 18,000 applications would have to be acquired and managed to cover demand. This would reduce total hiring cost by about 4.6m Euros. Accordingly, we seek to identify PIOS as process characteristics which are apt to reduce the probability of the defective cases described. Table 1 describes our results in this respect.

Table 1. Sample Case: Process Improvement Objectives

PIOs	Rationale
<i>Reducing processing cost</i>	Emerging potentials in terms of reducing process enactment effort per instance should be addressed.
<i>Reducing failed approvals</i>	Final approval of job offers by senior management fails if there are issues regarding vacancy management, reconciliation of terms, or checking of documents. The probability of these “late defects” should be addressed.
<i>Reducing cycle times</i>	The probability of applicants’ obtaining and taking alternative job offers increases with time. Therefore, cycle times between applications being received and job offers made should be as short as possible.

Note that for the first PIO, *Reducing processing cost*, there is an evident link to our organizational objective of reducing cost per hire. However, the second and third PIOS, *Reducing failed approvals* and *Reducing cycle times*, are based on hypotheses on how process enactment defects which affect the organizational objective can be reduced. Accordingly, they require qualitative or quantitative evidence with regard to their relevance in terms of reducing enactment defects and thus, in the end, improving cost per hire.

For the second PIO, *Reducing failed approvals*, we obtained qualitative evidence by interviewing responsible managers, which confirmed the underlying topics described in Table 1. Since the reasons for failed approvals are not captured in the applications management PAIS, quantitative evidence is not available. For the third PIO, *Reducing cycle times*, the causal link to its underlying defect of applications withdrawn by candidates is not as obvious. However, the correlation can be quantitatively demonstrated:

Correlation between Job Offers Declined and Cycle Times. We want to determine whether there is a significant influence of cycle time between application receipt and job offer in weeks on the probability of an applicant accepting or declining a job offer. Accordingly, we use a binary logistic regression test to evaluate the influence of a metric independent variable on a binary dependent variable. For the test, we use a sample of 2,721 job offers representing about 70% of the annual volume (cf. Fig. 2), and consisting of instances fully covered in the PAIS (not all interviews and feedbacks are documented in the PAIS). The sample contains 261 cases where the job offer was eventually declined by the applicant. This is the latest point in the process where withdrawal by the applicant is possible, and a significant amount of effort will have been spent on each respective case. Both independent samples have a size of more than 100 cases. Thus, the binary logistic regression can be applied. Fig. 6 shows an excerpt from the output of the statistical software package we used (Minitab). The p-value of

less than 0.05 indicates sufficient evidence to assume a significant impact of cycle time. According to the “Odds Ratio” column, a one week delay can be expected to increase the probability of an applicant declining a job offer by 16%.

Logistic Regression Table						
Predictor	Coef	SE	P-Value: Probability of duration not being a relevant factor	P	Odds Ratio	Odds Ratio: Lowering duration by one week expected to reduce withdrawal risk by 16%
Constant	-2.58986	0.169500	-15.26	0.000		
Duration_weeks	0.144378	0.0635831	2.27	0.023	1.16	1.02 1.31

Fig. 6. Minitab Output Excerpt: Binary Regression Test

Key Principle 4 (Considering Labor Relations and Restrictions in Quantitative Analyses). Quantitative analyses to support PIO and PIM validation like, for example, the use of process mining requires the collection of data on actual process enactment. However, it is not a research objective to assess individual performance of employees. In Germany, for example, intentions in this respect are subject to worker participation regulations according to the *Betriebsverfassungsgesetz* (the federal code on co-determination). Thus, researchers must be careful when designing quantitative analyses and the respective data collection procedures. Otherwise, organizations may refrain from providing relevant data.

4.3 Process Improvement Measures (PIMs)

To address the PIOs described in the previous section, relevant PIPs are bundled into PIMs specific to the application scenario. In our case, PIPs have been selected from a framework by Reijers & Liman Mansar [1] on process redesign practices (these are marked with an asterisk “*”) as well as from our ongoing research on improving business process quality [18,19]. Accordingly, as a mental technique to identify propositions applicable to our application scenario, we reflect available results on PIPs against our PIOs to obtain PIMs (cf. Fig. 3). Table 2 summarizes PIOs and corresponding PIMs as bundles of PIPs.

Key Principle 5 (Prospective and Retrospective Identification of PIOs and Potential PIPs). Relevant PIOs and potentially applicable PIPs should be identified not only by prospectively considering the process model, but also by retrospectively analyzing empirical data on process enactment. This is crucial to focus on topics of actual value potential. For example, consider the selection of critical cases in Fig. 2 that is reflected in the PIOs for our sample case.

Table 2. Defining Process Improvement Measures for Process Improvement Objectives

PIOs	Applicable PIMs with Comprised PIPs
<i>Reducing processing cost</i>	<i>PIM 1 (Application Management Automation)</i> : Task automation*, routing automation
<i>Reducing failed approvals</i>	<i>PIM 2 (Utilization and Capacity Management)</i> : Empowerment*, knock-out* <i>PIM 3 (Standardized Terms and Conditions)</i> : Triage*, buffering*
<i>Reducing cycle times</i>	<i>PIM 4 (Managing Interview Feedback Cycle Times for Successful Applicants)</i> : Control addition*, routing automation, escalation procedure <i>PIM 5 (Improving Application Routing)</i> : Case manager*, knock-out*, mitigation of repetitive loops

In actual design and implementation projects, it is common to document and track individual PIMs through *measure cards*. As an example of this practice, we choose two PIMs from Table 2 and describe them in more detail. For each PIM, we give a short content description – with PIPs involved marked as *italic* – and required implementation effort. On that basis, we appraise the business value considering the impact on PIOS as well as implementation effort. Results of our appraisal are validated through interviews with respective stakeholders.

Key Principle 6 (Considering Implementation Effort in Business Value Appraisal). When discussing the business value of particular PIMs for a business process, the respective implementation effort, including measures required, cost, time, and change management issues (e.g., training personnel to enact new activities), must be taken into account. Accordingly, a PIM will provide business value only if implementation effort is justified by realized process improvement potentials. For example, an organization may demand that the required investment may not exceed three times the projected annual cost savings when appraising operational cost optimization measures.

Note that, in addition to the scope presented here, actual *measure cards* comprise additional information relevant to project management such as project planning, project organization, key milestones with “traffic light” status, risks, next steps, and decision requirements. Reporting on measure cards usually takes place in steering committee meetings of senior management.

PIM Card 2 (Utilization and Capacity Management). Among other reasons, senior managers refuse to approve job offers when the business unit wishing to hire a candidate cannot fully utilize present capacity (this is a common performance indicator in professional services). While refusal reasons are not tracked in the PAIS, stakeholder interviews resulted in an estimate of about 30% of total

refusals to be caused by this issue. Since candidates' qualifications, in particular in graduate recruiting, are mostly not specific to particular business units, the recruiting department can be *empowered* to route applications to more appropriate teams from the start on. This results in an early *knock out* of applications that would, in the end, be declined because of low utilization.

Implementation. To enable utilization-based routing decisions, a new report on utilization per team must be integrated into the application management workflow. Since relevant data is available, and is routinely checked at other spots, the corresponding implementation effort has been estimated to be 25 consultant days or 27,500 Euros. In addition, relevant utilization thresholds must be agreed and communicated. The recruiting center routes about 12,000 applications per year. If the additional operating effort for the utilization check can be assumed to be 10 minutes per application, this results in an overall additional capacity requirement of about 1.2 full time equivalents (FTEs), resulting in approximately 84,000 Euros annual cost.

Business Value. The PIM is expected to reduce the "late refusal" rate by about 30% or 120 cases per year. Assuming a rate of job offers declined by the applicant of 7% (cf. Sect. 4.2), this would reduce the number of applications to be generated and managed to achieve a constant volume of hires by about 1,200. As we assumed the cost per application to be about 400 Euros, an annual savings potential of 480,000 Euros compares to 27,500 Euros one-off cost and 84,000 Euros operating expenditure per year.

Stakeholder Verification. When discussing the PIM with senior stakeholders, its business value appeared as rather clear. However, the distribution of utilization data emerged as a "political" issue. Considering present organizational culture, the PIM will not be implemented, but the basic capability to add utilization control functionality to the PAIS will be included with the requirements definition for the new PAIS solution to be completed by early 2013.

The abbreviated measure card presented above exemplifies how PIM implementation benefits can be projected and matched against expected implementation effort. However, beyond this quantitative reasoning, qualitative (or "political") topics can play a role in implementation decisions as well.

PIM Card 4 (Managing Interview Feedback Cycle Times for Successful Applicants). The time span between successful interviews and job offers can be reduced by implementing a *control addition*, i.e., additional control flow elements to ensure the correct enactment of the process. Triggered through *routing automation*, the recruiting department will call the interviewer directly when feedback is not available five business days after an interview. If the interviewer cannot be reached within two business days, an *escalation procedure* will take place by calling the respective supervisor. If no feedback can be obtained through these PIPs within ten business days, a letter of refusal will be sent.

Implementation. To implement the PIM, comprehensive tracing of interview dates and an additional workflow with corresponding triggering mechanisms

must be implemented in the PAIS. This results in an estimated cost of approx. 38,500 Euros for 35 consultant days. In the data sample used for the binary regression test in Sect. 4.2, about 51% of cases would fall under the proposed regulations. Accordingly, a total volume of 7,000 interviews conducted annually (cf. Fig. 2) would result in about 3,600 escalation procedures. On the one hand, this number can be expected to decline over time. On the other hand, multiple phone calls might be necessary for one escalation case. Hence, we assume that 20 escalation procedures can be handled in one person day. This means that an additional 0.9 FTEs are required, resulting in about 63,000 Euros annual cost.

Business Value. Based on our binary logical regression analysis (cf. Sect. 4.2), we reconciled with stakeholders that the maximum interview feedback time can be reduced to two weeks based on an escalation process. Applying the corresponding odds ratio (cf. Sect. 4.2) to all cases in our sample which exceed this timeframe results in a reduction of 39.2 cases of “late withdrawals” (cf. Fig. 2). This would reduce the number of applications to be generated and managed by about 390 per year, corresponding to 156,000 Euros in annual savings. Considering additional operating expenditures of 63,000 Euros results in a total annual cost reduction of 93,000 Euros versus a one-off cost of 38,500 Euros.

Stakeholder Verification. During stakeholder interviews, we validated implementation cost with the application workflow administrator, additional processing effort at the recruiting center with the head of recruiting, and overall viability of the new process with the head of recruiting and the business unit HR partner. The escalation procedure to provide timely feedback was challenged by the business unit HR partner, but not by team managers. Final consent on the positive business value of the PIM could be achieved by discussing the quantitative analysis of the underlying PIO (cf. Sect. 4.2).

Note that the PIMs presented exhibit a fairly positive business case with implementation cost to total annual savings ratios below two years. They constitute good examples of a phenomenon often encountered in practice: in many cases, it is interesting to first identify and resolve existing process defects within the framework of available technology before additional process automation is implemented at huge cost.

Key Principle 7 (Leveraging “Quick Win” Potentials). In many practical scenarios, it is possible to identify “quick win” PIMs that can be implemented with limited effort and should thus be prioritized, in particular in comparison to full-scale PAIS implementation measures which are typically very costly. Examples include the elimination of process defects caused by process participants’ behavior, interface issues between departments, or issues with respect to master data quality. Note that these topics are often identified through empirical analyses (e.g., using process mining).

5 Related Work

Approaches aiming at empirical validation of PIPs can be traced back to quality management and business process reengineering approaches which have evolved since the 1950s and the early 1990s, respectively.

In terms of quality management, Six Sigma [20] is of particular interest because it aims at eliminating errors in manufacturing processes. While the scope of BPM usually lies in administrative processes instead, there are interesting analogies since Six Sigma is based on step-by-step optimization of production processes through a-priori experimental changes to parameters.

Business process reengineering, as exemplified in [4,21], aims at optimizing processes “in the large” instead of implementing incremental PIPs. Transferring process enactment to an external supplier or customer constitutes a good example of this paradigm. While the potentials of this disruptive approach may seem tempting, later empirical research has shown that the risk of projects failing is substantial [22]. Thus, incremental improvement may still be a valid approach.

In contemporary BPM, [6] proposes a framework to select and implement redesign practices. As opposed to our research, this approach does not aim at evaluating individual PIPs, but at efficiently appraising a broad framework of practices. However, we use earlier results from the same authors as a source of PIPs to be assessed in more detail [1]. Note that research on PIPs addresses the quality of business processes in the sense of business content. In contrast, [23,24,25] exemplify propositions on process model quality in terms of structure, i.e., the proper partitioning of actual business content into model elements.

In IS research, there have been diverse propositions to ensure common standards of scientific rigor in empirical research such as field experiments or case studies [26,27]. As a basis of this paper, we chose the requirements summary by Wieringa et al. [15] due to its concise, checklist-based character, which makes it readily applicable to research as well as discussion with practitioners.

6 Conclusion

This paper described a methodology for a-priori, scenario-specific validation of process improvement patterns based on organizational objectives, process improvement objectives, and process improvement measures. In our methodology, we leveraged available work on generic requirements towards empirical research in IS engineering [15], thus demonstrating how these principles can be applied to practical cases while ensuring the general appeal of our approach.

We reported on the application of the methodology to a real-world business process, including validation of the respective results with practitioners. The organization hosting our sample application scenario is currently implementing a new application management PAIS to be completed in 2013. The agreed PIMs have been included in the requirements definition for this project.

In particular when discussing research design and execution with stakeholders, we identified a number of key principles useful as lessons learned for subsequent work which we included at appropriate spots in this paper.

References

1. Reijers, H.A., Liman Mansar, S.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega* 33(4), 283–306 (2005)
2. Heravizadeh, M., Mendling, J., Rosemann, M.: Dimensions of business processes quality (QoBP). In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008 Workshops. LNBIIP, vol. 17, pp. 80–91. Springer, Heidelberg (2009)
3. van der Aalst, W.M.P.: Reengineering knock-out processes. *Decision Support Systems* 30, 451–468 (2001)
4. Davenport, T.J., Short, J.E.: The new industrial engineering: Information technology and business process redesign. *Sloan Mgmt. Rev.* (4), 11–27 (1990)
5. zur Muehlen, M., Ho, D.T.: Service process innovation: A case study of BPMN in practice. In: Proc. 41st HICSS, p. 372. IEEE Computer Society (2008)
6. Liman Mansar, S., Reijers, H.A.: Best practices in business process redesign: use and impact. *Business Process Mgmt. J.* 13(2), 193–213 (2007)
7. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: Abstraction and reuse of object-oriented design. In: Wang, J. (ed.) ECOOP 1993. LNCS, vol. 707, pp. 406–431. Springer, Heidelberg (1993)
8. IDS Scheer: Process intelligence white paper: What is process intelligence? (2009), <http://www.process-intelligence.com>
9. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
10. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies. Springer (2012)
11. The Object Management Group: Business Process Model and Notation: Version 2.0 (January 2011), <http://www.omg.org/spec/BPMN/2.0>
12. Lohrmann, M., Reichert, M.: Demonstrating the effectiveness of process improvement patterns with mining results. Technical Report UIB-2013-03, Databases and Information Systems Institute, Ulm University (2013)
13. Simon, H.A.: The Sciences of the Artificial, 3rd edn. MIT Press (1996)
14. March, S.T., Smith, G.F.: Design and natural science research on information technology. *Decis. Support Syst.* 15(4), 251–266 (1995)
15. Wieringa, R., Heerkens, H., Regnell, B.: How to write and read a scientific evaluation paper. In: Proc. 17th RE, pp. 361–364. IEEE Computer Society (2009)
16. Best Management Practice: IT Infrastructure Library (2011)
17. Society for Human Resource Management: Executive brief: What factors influence cost-per-hire? (2012), <http://www.shrm.org/benchmarks>
18. Lohrmann, M., Reichert, M.: Understanding Business Process Quality. In: Glykas, M. (ed.) Business Process Management. SCI, vol. 444, pp. 41–74. Springer, Heidelberg (2013)
19. Lohrmann, M., Reichert, M.: Efficacy-aware business process modeling. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 38–55. Springer, Heidelberg (2012)
20. Pyzdek, T., Keller, P.: The Six Sigma Handbook. McGraw-Hill (2009)

21. Hammer, M., Champy, J.: Reengineering the corporation. A manifesto for business revolution. HarperBusiness (1993)
22. Cao, G., Clarke, S., Lehaney, B.: A critique of BPR from a holistic perspective. *Business Process Mgmt. J.* 7(4) (2001)
23. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of Business Process Modeling. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) *Business Process Management. LNCS*, vol. 1806, pp. 30–49. Springer, Heidelberg (2000)
24. Vanderfeesten, I., Reijers, H., van der Aalst, W.M.P.: Evaluating workflow process designs using cohesion and coupling metrics. *Comp. in Ind.* 59(5), 420–437 (2008)
25. Weber, B., Reichert, M., Mendling, J., Reijers, H.: Refactoring large process model repositories. *Comp. in Ind.* 62(5), 467–486 (2011)
26. Benbasat, I., Zmud, R.W.: Empirical research in information systems: The practice of relevance. *MIS Quarterly* 23(1), 3–16 (1999)
27. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting Experiments in Software Engineering. In: *Advanced Topics in Empirical Softw. Eng.* Springer (2007)

Enhancing Modeling and Change Support for Process Families through Change Patterns*

Clara Ayora¹, Victoria Torres¹, Barbara Weber², Manfred Reichert³,
and Vicente Pelechano¹

¹ Universitat Politècnica de València

{cayora, vtorres, pele} @pros.upv.es

² University of Innsbruck, Austria

barbara.weber@uibk.ac.at

³ University of Ulm, Germany

manfred.reichert@uni-ulm.de

Abstract. The increasing adoption of process-aware information systems (PAISs), together with the variability of business processes (BPs), has resulted in large collections of related process model variants (i.e., *process families*). To effectively deal with process families, several proposals (e.g., C-EPC, Provop) exist that extend BP modeling languages with variability-specific constructs. While fostering reuse and reducing modeling efforts, respective constructs imply additional complexity and demand proper support for process designers when creating and modifying process families. Recently, generic and language-independent adaptation patterns were successfully introduced for creating and evolving single BP models. However, they are not sufficient to cope with the specific needs for modeling and evolving process families. This paper suggests a complementary set of generic and language-independent change patterns specifically tailored to the needs of process families. When used in combination with existing adaptation patterns, *change patterns for process families* will enable the modeling and evolution of process families at a high-level of abstraction. Further, they will serve as reference for implementing tools or comparing proposals managing process families.

Keywords: Process Variability, Process Families, Patterns, Process Change.

1 Introduction

The increasing adoption of process-aware information systems (PAISs) has resulted in large process model repositories [25,6]. Since business process (BP) models usually may vary, existing repositories often comprise large collections of related *process model variants* (*process variants* for short) [24]. Usually, such process variants have common parts and pursue same or similar business objective, but at the same time

* This work has been developed with the support of MICINN under the Project EVERYWARE TIN2010-18011.

differ regarding the application context in which they are used [12,25], e.g., countries' regulations, services delivered, or customer categories [23,6,24]. We denote such collections of related process variants as *process families*. In large companies, a process family might comprise dozens or hundreds of process variants [23]. For example, a process family for vehicle maintenance may comprise more than 900 variants with country-, garage-, and vehicle-specific differences [13]. In turn, [21] reports on a process family comprising more than 90 variants for planning and handling medical examinations. Designing and implementing each process variant model from scratch and maintaining it separately would be too inefficient and costly. Thus, there is a great interest in capturing common process knowledge only once and re-using it in terms of *configurable process models* representing the complete process family.

Motivated by the shortcomings of traditional BP modeling approaches [13], proposals exist for dealing with process families, e.g., [26,13]. Common to them is the extension of BP modeling languages with variability-specific constructs that enable the creation of configurable process models. By treating variability as first class citizen, these extensions help avoiding redundancies, fostering reusability, and reducing modeling efforts. However, introducing variability-specific constructs implies additional complexity concerning the modeling language. To make these proposals amenable for industrial strength use, the quality of created models becomes crucial needing proper support for process designers when dealing with process families.

In [32], a language-independent and empirically grounded set of *adaptation patterns* is proposed allowing for the creation and modification of single BP models [31]. Adaptation patterns not only allow creating and modifying BP models at a high level of abstraction, fostering model quality by ensuring correctness-by-construction, but also provide systematic means for realizing change operations optimized for a specific modeling language as well as comparing existing approaches in respect to BP flexibility [7]. Further, adaptation patterns have served as basis for implementing changes in different stages of the process lifecycle; e.g., model creation [30,10], process configuration [13], process instance change [5,9,22], model evolution [5,17], model refactoring [33], change reuse [2], model comparison [16], and change analysis [11].

While adaptation patterns are well suited for creating and modifying single BP models, they are not sufficient to cope with the specific needs for dealing with process families [3]. In the vein of adaptation patterns, this paper suggests a complementary set of generic, language-independent change patterns specifically tailored for process families. Used in combination with the existing adaptation patterns, *change patterns for process families* will enable the modeling and evolution of process families at a high level of abstraction. In particular, they will serve as reference for specific language-dependent implementations, build the foundation for realizing changes along the BP lifecycle, and foster the comparison of existing proposals for BP variability.

Change patterns have been obtained after performing a systematic literature review looking specifically at variability-specific constructs used by existing proposals for BP variability. Since the proposed patterns are meant to be generic and language-independent, we abstract from approach-specific particularities. However, to ensure that the proposed patterns—despite their generic nature—are specific enough to cover

existing proposals, we apply them to two well-known proposals dealing with process families, i.e., C-EPC and Provop.

The remainder of the paper is structured as follows. Sect. 2 discusses related work and Sect. 3 presents the performed systematic literature review. In Sect. 4, we present the variability-specific language constructs obtained from the latter. Sect. 5 presents nine change patterns for process families. Sect. 6 provides a discussion and Sect. 7 concludes the paper.

2 Related Work

Closely related to our work is research on adaptation patterns, workflow patterns, and process variability.

Adaptation patterns (AP) [31] allow structurally changing process models using high-level change operations instead of low level change primitives (e.g., add or delete node). They can be applied along to the entire process lifecycle and do not have to be pre-planned, i.e., the region to which adaptation patterns may be applied can be chosen dynamically. Hence, adaptation patterns are well suited for realizing process changes at both build- and run-time. AP1 and AP2 allow inserting and deleting process fragments. Moving and replacing fragments is supported by AP3 (MOVE Process Fragment), AP4 (REPLACE Process Fragment), AP5 (SWAP Process Fragment), and AP14 (COPY Process Fragment). AP6 and AP7 allow adding or removing levels of hierarchy, AP8-AP12 support adaptations of control dependencies: embedding process fragments in loops (AP8), parallelizing (AP9) or embedding them in a conditional branch (AP10), and adding/removing control dependencies (AP11, AP12). Finally, AP13 allows changing transition conditions. This paper complements adaptation patterns, which cover the basic use cases for creating and modifying process models, with a set of patterns covering variability needs in process families.

Workflow patterns were introduced for analyzing the expressiveness of process modeling languages. Patterns cover different perspectives like control flow [1], data [27], resources [28], time [18], and exceptions [29,20]. Further, [10] describes a set of pattern compounds, similar to adaptation patterns, allowing for the context-sensitive selection and pattern composition during process modeling. However, these patterns are not sufficient for effectively modeling and modifying process families. They do not consider variability-specific constructs introduced by process families and hence are complementary to our change patterns.

Proposals dealing with BP variability exist for modeling, configuring [26, 13], and maintaining process families; e.g., [15] provides a set of language-specific operators to adapt process variants at runtime based on software product line concepts. In [7], a combination of workflow-, rule-, and event-modeling is presented to customize process variants for a given execution context. Unlike these proposals, change patterns provide language-independent means to model and evolve process families at a high level of abstraction. Finally, there are refactoring techniques [33] to remove redundancies among process variants in large process model repositories.

3 Research Methodology

The *goal* of this paper is to provide a set of generic and language-independent patterns for modeling and evolving process families. We first present the research methodology we employed for identifying these patterns. To ensure that the latter are expressive enough to deal with the specific needs of process families, as basis, we identified the set of variability-specific language constructs frequently used by existing proposals to capture the variability within a process family. More precisely, we conducted a *systematic literature review* (SLR) [14] using the following procedure: (1) formulation of the research question, (2) description of a search strategy for finding relevant papers, (3) identification of inclusion and exclusion criteria, and (4) analysis of the data obtained. The main research question to be answered by the SLR is “*What variability-specific language constructs are provided by existing proposals for modeling BP variability and process families respectively?*”. For this, we selected the following search string (considering different synonyms):

(‘process family’ OR ‘configurable process model’ OR ‘process model collection’ OR ‘reference process model’ OR ‘configurable workflow’) OR ‘process variant’ OR ‘business process variability’ OR (‘process configuration’ OR ‘process model configuration’)

This search string was applied to relevant data sources: ACM Digital Library, IEEE Xplore Digital Library, Science Direct - Elsevier, SpringerLink, Wiley Inter Science, World Scientific, and Google Scholar. Overall, these libraries include the proceedings of the most relevant conferences and journals in the area of BP management; e.g., *Data & Knowledge Engineering Journal*, *Information Systems Journal*, *Conference on Business Process Management (BPM)*, *Conference on Advanced Information Systems Engineering (CAiSE)*, and *Working Conference of Business Process Modeling, Development, and Support (BPMDS)*. As an additional data source, we considered the references of the identified papers.

A paper was included in the SLR (i.e., inclusion criterion) if and only if its title, abstract, and content is related to process families, either from a theoretical or practical perspective. On the contrary, papers were excluded (i.e., exclusion criterion) if their focus was not related to process families (e.g., software product lines). Papers describing the same proposal were removed and only the most complete version was included. We did not use any restriction concerning the publication date and only papers written in English were included. Finally, we only included proposals for which an implementation or evaluation exists.

Our SLR resulted in a total of 4948 papers, which were manually reviewed. In total, 25 papers passed this filtering and were further analyzed. To identify the language constructs commonly used in BP proposals (and serving as basis for our change patterns), we first create a list of candidate constructs relying on our experience with process families [4,31,33]. Then, we analyzed the 25 identified papers to find empirical evidence for our candidate variability-specific language constructs and iteratively

refined the initial list. Finally, only those constructs for which enough empirical evidence exists were included in the final list of variability-specific constructs.

Although proposals use different terminology and realize constructs in different ways, the SLR revealed that they essentially support the same language constructs for dealing with BP variability. We identified four variability-specific language constructs commonly shared by the 25 proposals: *configurable region*, *configuration alternative*, *context condition*, and *configuration constraint* (see Sect. 4.1 for details). Configurable regions are supported by 20 of the 25 proposals and configuration alternatives by 22 proposals. Context conditions are covered by 16 proposals while 15 proposals support the definition of configuration constraints. Additional language constructs we identified (e.g., *configurable region resolution time*) are only considered by few proposals (<3) and are therefore not included in our final list of variability-specific language constructs (for further details on the SLR see¹).

The final list of four variability-specific language constructs was then used as a basis for the change patterns, which constitute hence a solution for changing process families developed with existing proposals. Since the proposed patterns are meant to be generic and language-independent, we abstracted from approach-specific particularities (cf. Sect. 4). Thereby, we focused on the *control flow* perspective since the SLR showed that this is the perspective mostly addressed by existing proposals. To ensure that the proposed patterns—despite their generic nature—are specific enough to cover existing proposals, we applied the respective patterns to two well-known proposal dealing with process families (cf. Sect. 5).

4 Coping with Variability in Business Process Families

This section describes the variability-specific language constructs obtained from the SLR and introduces two representative proposals to show how the change patterns can be realized. For illustration purpose, we make use of the process carried out when checking-in at an airport. We chose this process since it shows a high degree of variability; e.g., variability occurs due to the type of check-in (e.g., online, or at a counter), which also determines the type of boarding card (e.g., electronic vs. paper-based). Other sources of variability include the type of passenger (e.g., unaccompanied minors requiring extra assistance) and the type of luggage (e.g., overweight luggage).

4.1 Coping with Variability in Business Process Families

The SLR described in Sect. 3 has revealed that the following language constructs are commonly used by existing proposals to capture variability (although their concrete realization might differ) in addition to standard process modeling constructs (e.g., activities and gateways). These language constructs form the basis of the *change patterns for process families* (see Sect. 5).

¹ <https://pros.webs.upv.es/bpvar/SLR/BPVariability.rar>

Language Construct LC1 (Configurable Region). A *configurable region* is a region in a configurable process model for which different configuration choices may exist depending on the application context, e.g., the airline offers different ways of obtaining the boarding cards depending on the check-in type: printing a boarding card at the airline desk, download an electronic boarding card, or obtaining it via mobile phone.

Language Construct LC2 (Configuration Alternatives). A *configuration alternative* is defined as a particular configuration choice that may be selected for a specific configurable region, e.g., there exist different types of boarding card: paper-based, electronic, or in the mobile phone.

Language Construct LC3 (Context Condition). A *context condition* defines the environmental conditions under which a particular configuration alternative of a configurable region shall be selected, e.g., passengers with overweight luggage pay a fee.

Language Construct LC4 (Configuration Constraint). A configuration constraint is defined as a (structural) restriction of the selection of configuration alternatives of the same or different configurable regions. Respective constraints are based on semantic restrictions to ensure the proper use of configuration alternatives, e.g., staff members need to be localized when unaccompanied minors are travelling.

4.2 Proposals Dealing with Process Families

The SLR described in Sect. 3 identified 25 proposals for dealing with process families. In the following, we describe two of them in more detail and explain how the obtained variability-specific language constructs have been realized by these proposals. Sect. 5 will then apply the identified change patterns to these two proposals to demonstrate that the proposed patterns are indeed generic. As representatives, we select two proposals that are (1) well established and highly cited, and (2) take fundamentally different approaches to realize the variability-specific language constructs. This way we want to ensure that the proposed patterns are general enough to cover very distinct proposals, but still specific enough to cover their essence.

C-EPC. A possible way of specifying a configurable process model is by means of *configurable nodes*. Modeling languages supporting this approach include, for example, C-EPC and C-YAWL [8]. Basically, these proposals extend an existing BP modeling language by adding configurable elements for explicitly representing variability in process families. In the following, we take C-EPC [26] as representative of this approach since it constitutes a well-known proposal. Fig. 1 illustrates the configurable process model as C-EPC for the check-in process. Configurable nodes are depicted with a thicker line. A configurable region (LC1) in C-EPC is specified by a process fragment of the configurable process model with exactly one entry and one exit (i.e., SESE fragment), and may take two different forms. First, the SESE fragment may consist of a *splitting configurable connector*, immediately followed by a set of branches representing configuration alternatives, and a *joining configurable connector*; i.e., the *configurable connectors* delimit the configurable region (e.g.,

Configurable region 2 in Fig. 1). Alternatively, the SESE fragment may consist of a *configurable function* (e.g., Configurable region 1 and 3 in Fig. 1), which may be configured as ON (i.e., the function is kept in the model), OFF (i.e., the function is removed from the model), or OPT (i.e., a conditional branching is included in the model deferring the decision to run-time). In turn, a configuration alternative (LC2) is specified by a SESE fragment which may be included as a branch between two *configurable connectors* (e.g., *Print electronic boarding card* in Configurable region 2 in Fig. 1). Context conditions (LC3) are represented in C-EPC separately in a *questionnaire model* [19], which is not considered in this paper. Finally, a configuration constraint (LC4) is specified by a *configuration requirement* linked to the configurable nodes that delimit the configurable region to which the respective configuration alternatives belong (e.g., *Configuration requirement 1* in Fig. 1 states that the inclusion of the function *Fill in UM form* implies the inclusion of the function *Localize staff*).

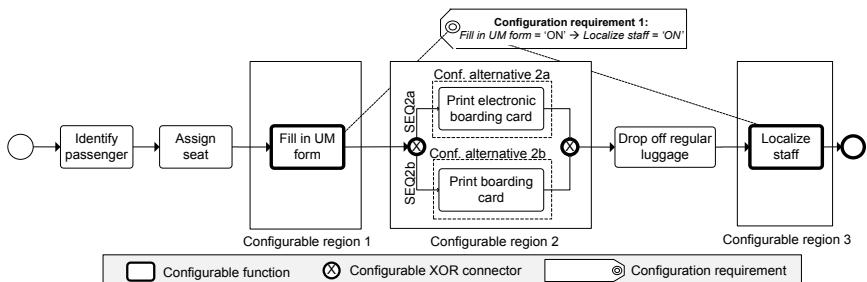


Fig. 1. C-EPC configurable process model for the check-in process

Provop. Another way of handling process families is based on the observation that process variants are often derived by adapting a pre-specified *base process model* (*base process*, for short) to the given context through a sequence of structural adaptations. The Provop proposal follows this approach [13]. We choose it since it provides advanced tool support for adapting a base process and for ensuring syntactical and semantical correctness of process variants derived. Fig. 2 illustrates how the process family dealing with the check-in process can be represented using Provop. The top of Fig. 2 shows the base process model from which the process variants may be derived. In Provop, a configurable region (LC1) is specified by a SESE fragment of the base process, delimited by two *adjustment points*; i.e., black diamonds (e.g., Configurable region 1 comprises the process fragment delimited by adjustment points A and B in Fig. 2). In turn, a configuration alternative (LC2) is specified by a *change option* that includes (1) the list of *change operations* modifying the base process at a specific configurable region and (2) a *context rule* that defines the context conditions under which the change operations shall be applied (e.g., Opt. 1 in Fig. 2). Context conditions (LC3) are specified by context rules which include a set of context variables

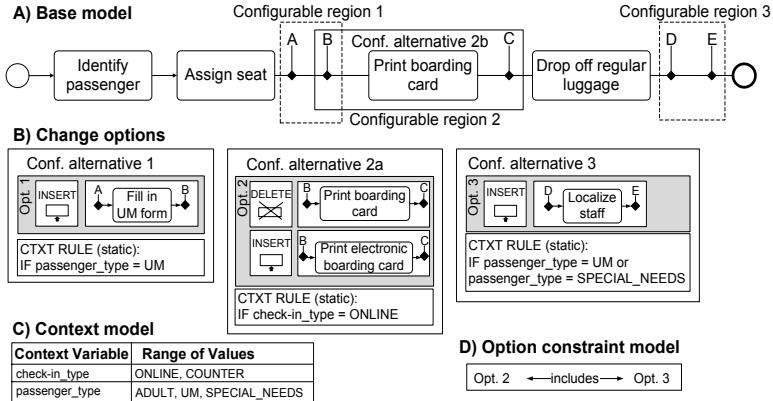


Fig. 2. Provop model for the check-in process

and their values specifying the conditions under which a configuration alternative (i.e., a change option) shall be applied (e.g., Opt. 2 is applied if the check-in type is online). All context variables and their allowed values are gathered all together in the *context model* (cf. Fig. 2C). Finally, configuration constraints (LC4) are specified as constraints (e.g., mutual exclusion) between two change options in the *option constraint model*; e.g., if Opt. 2 is applied then Opt. 3 has to be applied as well (cf. Fig. 2C).

5 Coping with Variability in Business Process Families

This section presents nine change patterns we consider as relevant for dealing with changes in process families. These patterns refer to the four variability-specific language constructs we obtained from our systematic literature review in existing proposals dealing with BP variability. Thus, proposed patterns support changes in process families developed with these proposals. Our change patterns are generic in the sense that they abstract from proposal-specific details. Moreover, they intend to be complete regarding the control flow perspective and cover all changes related to commonly used variability-specific language constructs. Further, we suppose that the change patterns will be combined with adaptation patterns to allow for the modeling and evolution of process families at a high level of abstraction. As illustrated in Table 1, we divide the change patterns into three categories: *insertion*, *deletion*, and *modification* of variability-specific parts of a configurable process model.

All change patterns, except CP7, allow adding (removing) variability-specific language constructs to (from) a configurable process model, representing the process family. In turn, pattern CP7 allows changing the conditions under which a configuration alternative is selected. To keep the pattern set minimal, we do not consider

patterns for *modifying* configurable regions, configuration alternatives, and configuration constraints. These modifications can be realized based on the combined use of change patterns and adaptation patterns. For example, modifying a configuration alternative may be implemented applying patterns CP3 and CP4, which, in turn, make use of respective adaptation patterns. Further, adding or removing process fragments which are shared by all process variants (i.e., commonalities), may be realized using adaptation patterns AP1 and AP2 (INSERT/DELETE Process Fragment).

Table 1. Change patterns for process families

CP1: INSERT Configurable Region
CP2: DELETE Configurable Region
CP3: INSERT Configuration Alternative IN a Configurable Region
CP4: DELETE Configuration Alternative IN a Configurable Region
CP5: INSERT Context Condition OF a Configuration Alternative
CP6: DELETE Context Condition OF a Configuration Alternative
CP7: MODIFY Context Condition OF a Configuration Alternative
CP8: INSERT Configuration Constraint BETWEEN Configuration Alternatives
CP9: DELETE Configuration Constraint BETWEEN Configuration Alternatives

Due to lack of space, we only present three change patterns related to the insertion of variability-specific constructs in more detail, i.e., CP1, CP3, and CP8 (cf. Figs. 3-6). The other change patterns are made available in a technical report [4]. For each of the change patterns, we provide a name, a brief description, an illustrative example, a description of the problem addressed, and corresponding design choices (indicating pattern variants). For example, CP1 presents three design choices (cf. Figs. 3-4): insert a configurable region as a new process region with a set of new configuration alternatives, inserting it by transforming a commonality into a configuration alternative (i.e., a common process fragment now is only applied in a specific application context), or by transforming a set of commonalities into a set of configuration alternatives. To demonstrate that the patterns—despite their generic nature—still cover the essence of different proposals for BP variability, we apply them to C-EPC and Provop, and show how they can be realized in their context. For example, regarding CP1, for each design choice, we indicate for both C-EPC and Provop how CP1 can be implemented using adaptation patterns. Further, note that for C-EPC we provide implementation details distinguishing between (i) *configurable functions* and (ii) *configurable connectors* since both allow representing configurable regions. In addition, we provide information about the parameters needed for each pattern. For example, realizing CP1 requires (1) the precise position in the configurable process model where the configurable region shall be inserted and (2) the configuration alternatives to be inserted in the configurable region (if needed). This information is highlighted in gray in the figures indicating how change patterns may be realized.

Pattern CP1: INSERT Configurable Region

Description: In a configurable process model, a configurable region shall be added.

Example: The way how boarding cards are handled depends on the type of check-in (e.g., paper-based vs. electronic cards). Assume that the configurable process model has not considered these alternatives yet. Hence, a new configurable region needs to be added.

Problem: At a certain position in the configurable process model, different configuration alternatives exist not reflected in the configurable process model so far. Hence, a configurable region covering these configuration alternatives shall be added.

Design choices: Three different design choices (DCs) exist:

DC1) Insertion as a new configurable region with up to n conf. alternatives ($n \geq 0$)

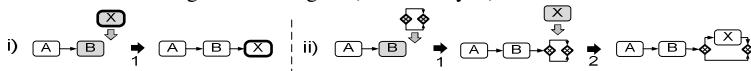
DC2) Insertion as a new configurable region by transforming a common process fragment into a configuration alternative

DC3) Insertion as a new configurable region by transforming existing process fragments into a set of configuration alternatives

Implementation in C-EPC:

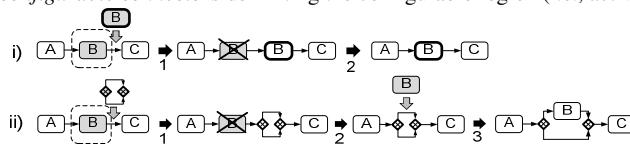
- For DC1, CP1 is realized by

1. applying adaptation pattern AP1 (i.e., *INSERT Process Fragment*) to insert the configurable region using either (i) a *configurable function* or (ii) two *configurable connectors* (i.e., split and join) at the precise position where the configurable region should be located (i.e., after activity B),
2. applying repeatedly CP3 (*INSERT Configuration Alternative IN a Configurable Region*) to insert a process fragment representing the configuration alternative (only relevant for *configurable connectors*), i.e., the configuration alternative is added as a branch between the two *configurable connectors* delimiting the conf. region (i.e., activity X).



- For DC2, CP1 is realized by

1. applying adaptation pattern AP1 (i.e., *INSERT Process Fragment*) to insert the configurable region using either (i) a *configurable function* or (ii) two *configurable connectors* (i.e., split and join) at the precise position where the configurable region should be located (i.e., after activity B),
2. applying adaptation pattern AP2 (i.e., *DELETE Process Fragment*) to delete the common process fragment from its current position (i.e., activity B), and
3. applying CP3 (*INSERT Configuration Alternative IN a Configurable Region*) to re-insert the common process fragment as a configuration alternative of the configurable region (only relevant for *configurable connectors*), i.e., the configuration alternative is added as a branch between the two *configurable connectors* delimiting the configurable region (i.e., activity B).

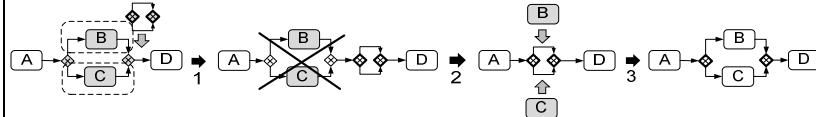


- For DC3, CP1 is realized by

1. applying adaptation pattern AP1 (i.e., *INSERT Process Fragment*) to insert the configurable region (only relevant for *configurable connectors*) at the precise position where the configurable region should be located (i.e., after the join XOR gateway),
2. applying adaptation pattern AP2 (i.e., *DELETE Process Fragment*) to delete the existing process fragment from its current position, and

Fig. 3. CP1 (INSERT Configurable Region)

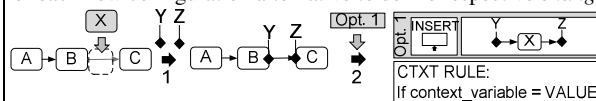
3. applying repeatedly CP3 (*INSERT Configuration Alternative IN a Configurable Region*) once per configuration alternative to re-insert the existing process fragments as configuration alternatives of the configurable region, i.e., each branch of the process fragment is added as a branch between the two *configurable connectors* delimiting the configurable region (i.e., activity B is inserted as one alternative and activity C as another one).



Implementation in ProVOp:

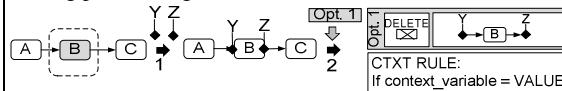
- For DC1, CP1 is realized by

1. inserting two *adjustment points* (i.e., Y and Z) in the *base process* and
2. applying repeatedly CP3 (*INSERT Configuration Alternative IN a Configurable Region*) once for each new configuration alternative to define respective *change options* (i.e., Opt. 1).



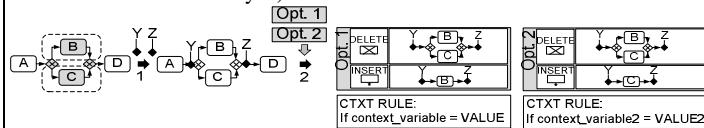
- For DC2, CP1 is realized by

1. inserting two *adjustment points* (i.e., Y and Z) embedding an existing process fragment of the *base process* (i.e., activity B) and
2. applying CP3 (*INSERT Configuration Alternative IN a Configurable Region*) to define a conf. alternative in terms of a *change option* inserting the existing process fragment into/removing the existing process fragment under certain conditions from the *base process* (i.e., Opt. 1).

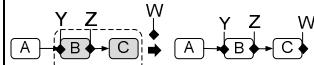


- For DC3, CP1 is realized by

1. inserting two *adjustment points* (i.e., Y and Z) embedding an existing process fragment of the *base process* (i.e., the process fragment becomes optional) and
2. applying repeatedly CP3 (*INSERT Configuration Alternative IN a Configurable Region*) to define the set of configuration alternatives in terms of *change options* inserting/removing existing process fragments under certain conditions from the *base process* (i.e., one option for activity B and another one for activity C).



If *adjustment points* already exist at the entry or exit of the new configurable region (e.g., as part of another configurable region) these may be reused instead.



Parameters:

- the position in the configurable process model where the configurable region shall be inserted
- the configuration alternative(s) to be added to the configurable region

Fig. 3. (continued)

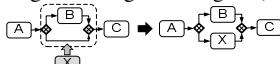
Pattern CP3: INSERT Configuration Alternative IN a Configurable Region

Description: In a configurable process model, a configuration alternative shall be added to a specific configurable region.

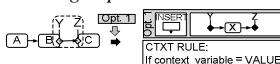
Example: Assume that the airline now wants to offer the possibility of obtaining the boarding card for smart phones as well. Thus, an alternative shall be added to this configurable region that captures how boarding cards are obtained.

Problem: For a specific configurable region of the configurable process model, existing config. alternatives do not cover all possible choices and hence an additional one shall be inserted.

Implementation in C-EPC: CP3 is realized by applying adaptation pattern AP1 (i.e., *INSERT Process Fragment*) to insert the process fragment representing the configuration alternative, i.e., the configuration alternative is added as a branch between the two *configurable connectors* delimiting the configurable region (i.e., activity X).



Implementation in Provol: CP3 is realized by defining a *change option* consisting of a sequence of *change operations* and a *context rule*.



Parameters:

- the configurable region to which the configuration alternative belongs
- the configuration alternative to be inserted

Fig. 4. CP3 (INSERT Configuration Alternative IN a Configurable Region)

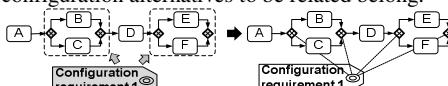
Pattern CP8: INSERT Configuration Constraint BETWEEN Configuration Alternatives

Description: In a configurable process model, a constraint regarding the use of configuration alternatives from one or more configurable regions shall be added.

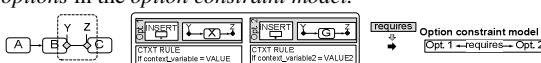
Example: When unaccompanied minors are travelling, extra staff is required to accompany them to the boarding gate, i.e., an inclusion constraint exists.

Problem: The use of alternatives needs to be constrained in a configurable process model.

Implementation in C-EPC: CP8 is realized by inserting a *configuration requirement*, which is then linked to the configurable nodes that delimit the configurable region to which the respective configuration alternatives to be related belong.



Implementation in Provol: CP8 is realized by adding a constraint regarding the use of *change options* in the *option constraint model*.



Parameters:

- the configuration region to which the alternatives whose use shall be constrained belong
- the configuration constraint to be inserted

Fig. 5. CP8 (INSERT Configuration Constraint BETWEEN Configuration Alternatives)

6 Discussion

Even though—as shown by the systematic literature review—existing proposals use different terminology and realize the constructs in different ways, they essentially support the same variability-specific language constructs. Similar to adaptation patterns [31], change patterns may have the potential to speed up the creation as well as modification of configurable process models. In addition, like adaptation patterns, the *change patterns for process families* may therefore serve as benchmark for evaluating change support in existing languages and tools dealing with process families as well as for facilitating their systematic comparison by providing a frame of reference. To substantiate these claims, we plan to conduct empirical studies testing the impact of the proposed patterns on both the creation and evolution of configurable process models. Moreover, in a similar vein than adaptation patterns, the proposed change patterns may serve as a reference for realizing changes in different stages of the process family life cycle, e.g., modeling, maintenance, and evolution.

As with every research, our work is subject to limitations. A first one concerns the completeness of the proposed patterns. We tried to accommodate this by grounding patterns on a SLR covering 25 different proposals for process families and by using variability-specific language requirements commonly occurring as basis for our patterns. As a consequence, the proposed pattern set intends to be complete in the sense that it allows modeling and modifying process families according to existing proposals dealing with BP variability, covering all possible changes related to commonly used variability-specific language constructs. However, we cannot state with certainty that the identified patterns set is sufficiently large to address all potential use cases regarding the modeling and change of process families in the most efficient way. For this, further empirical studies on the practical use of the patterns are needed. Closely related to this are considerations regarding the language-independent nature of the proposed patterns. Using commonly occurring variability-specific constructs as a basis, we can ensure that the proposed patterns are expressive enough to model and modify process families. To ensure that the patterns are also specific enough to cover the particularities of the different proposals, we applied them to two commonly used and entirely different proposals for process families. To strengthen the validation of the patterns, they will be applied to other proposals in future work. Moreover, the focus of the proposed patterns is currently on variability-specific constructs regarding the *control flow* perspective. Variability regarding additional perspectives like data or resources has not been considered so far.

The proposed patterns have been described in an informal way. To obtain unambiguous pattern descriptions and ground pattern implementation as well as pattern-based analysis on a sound basis, a formal semantics is needed. This formalization should be independent from any process meta model and thus allow implementing the patterns in a variety of process support tools.

7 Conclusions and Outlook

We proposed nine patterns for dealing with changes in process families. We complement existing work on patterns for creating and modifying BP models by introducing a set of generic and language-independent patterns that cover the specific needs of process families. The patterns are based on variability-specific language constructs obtained from a systematic literature review. To demonstrate that they still cover the essence of existing proposals managing BP variability, we applied them to two representative proposals. Used in combination with adaptation patterns, change patterns for process families allow modeling and evolving process families at an abstract level. In future work, we will develop a prototype based on which we will conduct experiments to measure the efforts of handling variability in process families. We will study the impact of patterns on modeling process families as well as on changing either at design or run-time.

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M., Barros, B.: Workflow Patterns. *Distributed and Parallel Databases* 14(1), 5–51 (2003)
2. Aghakasiri, Z., Mirian-Hosseiniabadi, S.H.: Workflow change patterns: Opportunities for extension and reuse. In: Proc. SERA 2009, pp. 265–275 (2009)
3. Ayora, C., Torres, V., Reichert, M., Weber, B., Pelechano, V.: Towards run-time flexibility for process families: Open issues and research challenges. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 477–488. Springer, Heidelberg (2013)
4. Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: Change patterns for process families. Technical Report, PROS-TR-2012-06, <http://www.pros.upv.es/technicalreports/PROS-TR-2012-06.pdf>
5. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. *Com. Sci. - R&D* 23, 81–97 (2009)
6. Dijkman, R., La Rosa, M., Reijers, H.A.: Managing large collections of business process models - Current techniques and challenges. *Comp. in Ind.* 63(2), 91–97 (2012)
7. Döhring, M., Zimmermann, B., Karg, L.: Flexible workflows at design- and runtime using BPMN2 adaptation patterns. In: Abramowicz, W. (ed.) BIS 2011. LNBIP, vol. 87, pp. 25–36. Springer, Heidelberg (2011)
8. Gottschalk, F.: Configurable process models. Ph.D. thesis, Eindhoven University of Technology, The Netherlands (2009)
9. Grambow, G., Oberhauser, R., Reichert, M.: Contextual injection of quality measures into software engineering processes. *Intl. J. Adv. in Software* 4, 76–99 (2011)
10. Gschwind, T., Koehler, J., Wong, J.: Applying patterns during business process modeling. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 4–19. Springer, Heidelberg (2008)
11. Günther, C.W., Rinderle, S., Reichert, M., van der Aalst, W.M.P.: Change mining in adaptive process management systems. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 309–326. Springer, Heidelberg (2006)
12. Hallerbach, A., Bauer, T., Reichert, M.: Context-based configuration of process variants. In: Proc. TCoB 2008, pp. 31–40 (2008)
13. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the ProVop approach. *J. of Software Maintenance* 22(6-7), 519–546 (2010)

14. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE/EPIC-2007-01 (2007)
15. Kulkarni, V., Barat, S., Roychoudhury, S.: Towards business application product lines. In: France, R.B., Kazmeier, J., Breu, R., Atkinson, C. (eds.) MODELS 2012. LNCS, vol. 7590, pp. 285–301. Springer, Heidelberg (2012)
16. Küster, J.M., Gerth, C., Förster, A., Engels, G.: Detecting and resolving process model differences in the absence of a change log. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 244–260. Springer, Heidelberg (2008)
17. Küster, J.M., Gerth, C., Engels, G.: Dynamic computation of change operations in version management of business process models. In: Kühne, T., Selic, B., Gervais, M.-P., Terrier, F. (eds.) ECMFA 2010. LNCS, vol. 6138, pp. 201–216. Springer, Heidelberg (2010)
18. Lanz, A., Weber, B., Reichert, M.: Time patterns for process-aware information systems. Requirements Engineering, 1–29 (2012)
19. La Rosa, M., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-based variability modeling for system configuration. Software and System Modeling 8(2), 251–274 (2009)
20. Lerner, B.S., Christov, S., Osterweil, L.J., Bendraou, R., Kannengiesser, U., Wise, A.: Exception Handling Patterns for Process Modeling. IEEE Transactions on Software Engineering 36(2), 162–183 (2010)
21. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: Challenges, scenarios, algorithms. Data Knowledge & Engineering 70(5), 409–434 (2011)
22. Marrella, A., Mecella, M., Russo, A.: Featuring automatic adaptivity through workflow enactment and planning. In: Proc. CollaborateCom 2011, pp. 372–381 (2011)
23. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 368–377. Springer, Heidelberg (2006)
24. Reichert, M., Weber, B.: Enabling flexibility in process-aware information systems: challenges, methods, technologies. Springer (2012)
25. Reinhartz-Berger, I., Soffer, P., Sturm, A.: Organizational reference models: supporting an adequate design of local business processes. IBPIM 4(2), 134–149 (2009)
26. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modeling language. Information Systems 32(1), 1–23 (2007)
27. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow data patterns. Technical Report FIT-TR-2004-01, Queensland Univ. of Technology (2004)
28. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow resource patterns. Technical Report WP 127, Eindhoven Univ. of Technology (2004)
29. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Workflow Exception Patterns. In: Martinez, F.H., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg (2006)
30. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Object-sensitive action patterns in process model repositories. In: Muehlen, M.z., Su, J. (eds.) BPM 2010 Workshops. LNBP, vol. 66, pp. 251–263. Springer, Heidelberg (2011)
31. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - Enhancing flexibility in process-aware information systems. Data Knowledge & Engineering 66, 438–466 (2008)
32. Weber, B., Sadiq, S., Reichert, M.: Beyond rigidity - dynamic process lifecycle support. Computer Science 23, 47–65 (2009)
33. Weber, B., Reichert, M., Reijers, H.A., Mendling, J.: Refactoring large process model repositories. Computers in Industry 62(5), 467–486 (2011)

Change Patterns in Use: A Critical Evaluation*

Barbara Weber¹, Jakob Pinggera¹, Victoria Torres², and Manfred Reichert³

¹ University of Innsbruck, Austria

{barbara.weber,jakob.pinggera}@uibk.ac.at

² Universitat Politècnica de València, Spain

vtorres@pros.upv.es

³ University of Ulm, Germany

manfred.reichert@uni-ulm.de

Abstract. Process model quality has been an area of considerable research efforts. In this context, the correctness-by-construction principle of change patterns provides promising perspectives. However, using change patterns for model creation imposes a more structured way of modeling. While the process of process modeling (PPM) based on change primitives has been investigated, little is known about this process based on change patterns. To obtain a better understanding of the PPM when using change patterns, the arising challenges, and the subjective perceptions of process designers, we conduct an exploratory study. The results indicate that process designers face little problems as long as control-flow is simple, but have considerable problems with the usage of change patterns when complex, nested models have to be created. Finally, we outline how effective tool support for change patterns should be realized.

Keywords: Process Model Quality, Process of Process Modeling, Change Patterns, Exploratory Study, Problem Solving.

1 Introduction

Much conceptual, analytical, and empirical research has been conducted during the last decades to enhance our understanding of conceptual modeling. In particular, process models have gained significant importance due to their fundamental role for process-aware information systems [1]. Even though it is well known that a good understanding of a process model has a direct and measurable impact on the success of any modeling initiative [2], process models display a wide range of quality problems impeding their comprehensibility and hampering their maintainability [3,4]. Literature reports, for example, on error rates between 10% and 20% in industrial process model collections [3].

To improve process model quality, change patterns offer a promising perspective [4]. They have well-defined semantics [5] and combine change primitives (e.g., to add nodes or edges) to high-level change operations. Particularly appealing is correctness-by-construction [6,7], i.e., the modeling environment only

* This research is supported by Austrian Science Fund (FWF): P23699-N23

provides those change patterns to the process designers, which ensure that a sound process model is transformed into another sound process model.

The use of change patterns implies a different way of creating process models, since correctness-by-construction imposes a structured way of modeling by enforcing block structuredness. Irrespective of whether change patterns or change primitives are used, model creation requires process designers to construct a mental model (i.e., *internal representation*) of the requirements to be captured in the process model [8]. In a subsequent step, the mental model is mapped to the constructs provided by the modeling language creating an *external representation* of the domain [8]. While the construction of the mental model presumably remains unaffected, the use of change patterns leads to different challenges concerning pattern selection and combination for creating the external representation. Further, the exact set of change patterns available influences the selection. While a large set of change patterns allows for more flexibility, it also increases complexity, making the modeling environment more difficult to use. Consequently, process designers might have to look several steps ahead to construct a certain process fragment, which constitutes a major difference compared to the use of change primitives, which do not impose any structural restrictions (i.e., no order is imposed when placing elements on the modeling canvas).

The process of creating process models based on change primitives has caused significant attention leading to a stream of research on the *process of process modeling* (PPM) [8,9,10,11]. This research is characterized by its focus on the formalization phase of model creation, i.e., the designer's interactions with the modeling environment [9]. Still, little is known about the PPM when utilizing change patterns. In this paper, we try to obtain an in-depth understanding of the PPM when using change patterns. In particular, this paper aims at understanding whether the necessity to look ahead leads to additional barriers during model creation and at shedding light on the challenges process designers face when using change patterns, which is essential to provide effective tool support.

To obtain an in-depth understanding of these issues, we implement a change pattern modeling editor based on Cheetah Experimental Platform (CEP) and conduct an exploratory study comprising several modeling tasks. The results of our study underline the potential for model creation based on change patterns. In particular, process designers did not face any major problem when using change patterns for constructing simple process fragments. When more complex process fragments forced process designers to look ahead, difficulties increased observably. Insights obtained from this exploratory study provide a better understanding of the PPM when using changes patterns and reveal challenges arising in such a context. In particular, results provide a contribution toward a better understanding on how tool features like change patterns impact the PPM, but also give advice on how effective tool support should be designed.

Sect. 2 introduces backgrounds. Sect. 3 describes the design of the exploratory study. Sect. 4 discusses challenges in change pattern use and Sect. 5 presents results on process designers' perception of patterns use. Results are discussed in Sect. 6. Related work is presented in Sect. 7. Sect. 8 concludes the paper.

2 Backgrounds

This section briefly introduces change patterns and then presents cognitive foundations and backgrounds of the PPM.

2.1 Process Modeling Based on Change Patterns

Change patterns provide a different way of interacting with the modeling environment in modeling phases. Instead of specifying a set of change primitives, the process designer applies change patterns (i.e., high-level change operations) to realize the desired model adaptation (cf. Fig. 1). Examples of change patterns include the insertion and deletion of process fragments, or their embedding in loops (the whole catalogue can be found in [12,4] and their semantics in [5]). When applying a single change primitive, soundness of the resulting process model cannot be guaranteed, but must be explicitly checked after applying the change primitives. On the contrary, change pattern implementations often associate pre-/post-conditions with high-level change operations to guarantee model correctness after each adaptation [6,7]. Process editors applying the correctness-by-construction principle (e.g., [12]) usually provide only those change patterns to the process designer that allow transforming a sound process model into another sound one. This is realized by imposing structural restrictions on process models (e.g., block structuredness).

- A) <Insert Node XOR-Split, Insert Control Edge, Insert Node XOR-Join>
- B) <Embed Process Fragment in Conditional Branch>

Fig. 1. Set of Change Primitives (A) and Patterns (B) to make an activity optional

2.2 Cognitive Foundations of Problem Solving

We consider the creation of process models to be a complex problem solving task. Problem solving has been an area of vivid research in cognitive psychology for decades. Therefore, we turn to cognitive psychology to understand the processes followed by humans when solving a problem like creating a process model.

Schemata. A central insight from cognitive research is that the human brain contains specialized regions contributing different functionality to the process of *solving complex problems*. *Long-term memory* is responsible for permanently storing information and has an essentially unlimited capacity, while in *working memory* comparing, computing and reasoning take place [13]. Although the latter is the main working area of the brain, it can store only a limited amount of information, which is forgotten after 20–30 seconds if not refreshed [14]. The question arises how information can be processed with such limited capacity. The human mind organizes information in interconnected *schemata* rather than in isolation [13]. Those schemata, stored in long-term memory, incorporate general concepts of similar situations [13]. Whenever situations similar to a schema

arise, the latter is retrieved to help organizing information by creating *chunks* of information that can be processed efficiently [15]. For instance, when asked to create a process model using change patterns, a designer must create an internal representation of the problem [8]. For this, information about the domain is retrieved and organized in memory using existing schemata for process modeling. Therefore, schemata guide the comprehension process, helping to re-organize information for its processing in working memory [15].

Problem-Solving Strategies. When confronting novices with an unfamiliar problem they cannot rely on specialized problem solving strategies. Instead, they must find a way to solve the problem and come up with an initial skeletal *plan* [16]. Then, novices utilize general problem solving strategies, like means-ends analysis, due to the lack of more specific strategies for the task at hand [17]. Means-ends analysis can be described as the continual comparison of the problem's current state with the desired end product. Based on this, the next steps are selected until a satisfying solution is found [17]. After applying the constructed plan, it can be stored in long-term memory as *plan schema* [16]. For this, task-specific details are removed from the plan schema resulting in a plan schema that can be automatically applied in similar situations [18]. When confronted with a problem solving task in the future, the appropriate plan schema is selected using a case-based reasoning approach [19]. The retrieved plan schema provides the user with structured knowledge that drives the process of solving the problem [15,19]. Plan schemata allow experts to immediately decide what steps to apply to end up with the desired solution [20]. If the plan schema is well developed, an expert never reaches a dead end when solving the problem [21].

Plan schemata seem to be important when creating process models based on change patterns since change pattern cannot be combined in an arbitrary manner. If no plan schema is available on how to combine change patterns to create the desired process model, designers have to utilize means-ends analysis until a satisfying solution is found. This behavior is more likely to reach in detours, reducing the process designer's efficiency when creating process models.

2.3 The Process of Process Modeling

During the formalization phase, process designers create a syntactically correct process model reflecting a given domain description by interacting with the process editor [22]. The formalization, i.e., the PPM, can be described as a cycle of the three phases of comprehension, modeling and reconciliation [9,8].

Comprehension. In comprehension phases designers try to understand the requirements to be modeled by extracting information from the task description and the existing process model to build an internal representation of the problem in working memory [17,21]. Depending on the availability of schemata for organizing the acquired knowledge, working memory is utilized more or less efficiently. If the process designer has solved a similar problem previously (i.e., a plan schema for the problem is stored in long-term memory), he can directly create the process model without any further attention on which steps to execute or

plan next. Either way, working memory is filled with knowledge extracted from requirements and, if available, the process model being created.

Modeling. The designer uses the internal representation developed during the comprehension phase to materialize the solution in a process model (by creating or changing it) [8,9]. Hence, modeling phases consist of a set of structural model adaptations. More specifically, designers interact with the modeling environment using change primitives such as `add activity` or `add edge`. The materialization of a process model adaptation may require the joint application of several change primitives. For instance, making an activity optional may require the application of three change primitives as shown in Fig. 1A. Once all information stored in working memory is incorporated in the process model, the designer interrupts the modeling endeavor to incorporate additional requirements into the internal representation [9].

Reconciliation. Designers may reorganize a process model (e.g., renaming of activities) and utilize its *secondary notation* (e.g., notation of layout, typographic cues) to enhance understandability [23]. However, the number of reconciliation phases in the PPM depends on the designer's ability of placing elements correctly when creating them [9].

3 Exploratory Study

To gain a better understanding of the PPM using change patterns we conduct an exploratory study. This section describes research questions and study design.

Research Questions. Central aim is to obtain an in-depth understanding of the PPM when using change patterns. More specifically, we try to understand the challenges designers are facing when using change patterns for model creation. Respective challenges can result in detours for designers on their way to a complete process model. Detours during process modeling result in decreased problem solving efficiency. Moreover, they might lead to modeling errors that persist in the final model.

RQ1: What are re-occurring challenges in the usage of change patterns that designers face and where do these challenges originate from?

The study also aims at investigating how designers experience their interaction with the modeling environment to create process models using change pattern.

RQ2: What is the subjective perception of designers when using change patterns for model creation?

Exploratory Study Execution. The design of the study consists of three phases. In the first phase, demographic data is collected. In the second phase, two modeling tasks are executed. When working on the modeling tasks all interactions with the modeling environment are recorded using CEP [24].

This allows us to replay the creation of the process model step-by-step (cf. [24,9]), addressing RQ1. After completing the modeling tasks, *Perceived Ease of Use* and the *Perceived Usefulness of Technology Acceptance Model (TAM)* [25] are assessed to investigate RQ2. In addition, participating subjects are asked to provide feedback regarding their experiences.

Subjects. The exploratory study was conducted in Innsbruck with 16 students of a graduate course on business process management. Since designers in practical settings are often not expert designers, but rather casual designers that only obtained a basic amount of training [26], we did not require modeling experts for our study. Subjects obtained basic training in modeling business processes prior to the exploratory study. In addition, they were taught theoretical backgrounds on change patterns prior to the exploratory study, but did not have any hands-on experience in the creation of process models using change patterns.

Modeling Tasks. For Task A, subjects received an informal requirements description and the solution of the modeling task (i.e., a process model). Subjects had to re-model the process using change patterns. Since subjects had the correct solution available, the challenge lies in determining the patterns to use for re-building the model and how to combine them effectively. This allowed students to develop problem solving strategies for utilizing change pattern. Task A was a process run by the “Task Force Earthquakes” of the German Research Center for Geosciences [27]. Subjects were asked to model the “Transport of Equipment” process using change patterns. The task requires sequences as well as conditional/parallel branchings. The solution model has a nesting depth of 2.

In Task B, designers had to create a process model starting from an informal description. This time, no solution model was made available to the subjects. Consequently, they not only had to decide which patterns to use and how to combine them, but additionally had to develop an understanding of the domain (by creating a mental model) and map it to the available change patterns. Therefore, schemata for extracting information from the textual description were necessary for completing the modeling task (e.g., for identifying activities). Task B describes the pre-take off procedures for a general aviation flight under visual rules [28]. Like Task A, it comprises sequences and conditional/parallel branchings. In terms of complexity, it only has a nesting depth of 1.

Change Pattern Set. When devising a modeling environment for model creation based on change patterns the question arises which change pattern set to provide. While a large set offers more flexibility, it also increases complexity—especially when mapping the mental model to the available patterns. We therefore utilize a minimal change pattern set (for the full pattern set see [4]), which allows designers to create all main control-flow constructs for tasks A and B (i.e., sequences, parallel branches, conditional branchings, and loops). The following patterns were available to the designers: AP1 (Insert Process Fragment), AP2 (Delete Process Fragment), AP8 (Embed Fragment in Loop), AP10 (Embed Process Fragment in Conditional Branch), and AP13 (Update Condition). Concerning AP1, two pattern variants were provided: Serial Insert and Parallel Insert.

4 Challenges in the Usage of Change Patterns

This section describes the data analysis procedure applied to obtain an in-depth understanding of re-occurring challenges during the PPM based on change patterns, answering RQ1 it further presents obtained results for Tasks A and B.

4.1 Data Analysis Procedure

Step 1: Determine Solution Model, Distance, and Optimal Problem Solving Paths. In a first step, for each modeling task, we create a model representing the correct solution (i.e., S_S). In a next step, we establish the *distance* for transforming an empty model (i.e., S_0) to S_S (i.e., the minimum number of change patterns needed). Typically, the process designer has several possibilities to create a solution model S_S by starting from S_0 and applying a sequence of model transformations. From a cognitive perspective, each possible sequence of change patterns that leads without detours to the correct solution constitutes an *optimal problem solving path*. For example, inserting region R4 of Task B in Fig. 2B starting from an empty modeling canvas can be achieved in different ways with a minimum number of 4 change patterns (e.g., S' can be created by first inserting $R4.1$, next embedding $R4.1$ in a conditional branch, then inserting $R4.2$, and finally updating the transition condition).

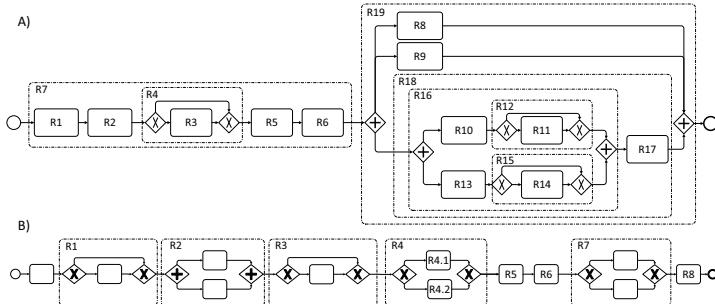


Fig. 2. Solution Models for Tasks A and B

Step 2: Determine Deviations from Optimal Problem Solving Path. To identify potential challenges designers were facing we analyze their problem solving paths for both modeling tasks using the replay functionality of CEP. For this, for each process designer we compare the problem solving path $P_{0,S}$ (i.e., sequence of patterns to transform S_0 to S_S) and capture deviations from the *optimal problem solving path*. Respective deviations can be detours the designer takes until coming up with the correct solution. Deviations quantify how efficient the chosen problem solving strategy is—denoted as *process deviations*. However, deviations can also be discrepancies between the model created by designers and the solution model S_S , denoted as *product deviations*. Fig. 3 shows the problem solving path of one process designer, who managed to model region R4 of Task B in

$$P_{0,S} = \langle \text{Serial Insert (R4.1)}, \text{Embed in Conditional Branch (R4.1)}, \\ \text{Serial Insert (R4.2)}, \text{Undo Serial Insert (R4.2)}, \\ \text{Serial Insert (R4.2)}, \text{Update Condition} \rangle$$

Fig. 3. Problem Solving Path with 2 Process Deviations

Fig. 2B correctly (i.e., 0 product deviations), but made a detour of 2 change patterns (crossed out lines) before reaching the solution (i.e., the solution path $P_{0,S}$ comprises 2 superfluous change patterns summing up to 2 process deviations).

Step 3: Classification of Deviations and Aggregation of Deviations. In Step 3, using the replay functionality of CEP, deviations are mapped to regions of the process model and reasons for every deviation are identified (e.g., misinterpretation of the textual description, problems with usage of patterns) in an iterative consensus-building process [29]. Moreover, to obtain an overview of which model parts caused most difficulties we aggregate for each task deviations per region.

4.2 Results Related to Task A

Regarding Task A, a minimum of 18 operations is needed to create the correct solution model (cf. Fig. 2A). Overall, we identified 254 deviations—232 process deviations (i.e., detours in the modeling process) and 22 product deviations (i.e., deviations of the final models from the solution model) (cf. Table 1). Process deviations per process designer ranged from 0 to 58, with an average of 13.4 deviations. In turn, product deviations ranged from 0 to 8 per process designer, with an average of 1.3 deviations.

Table 1. Overview of Results for Task A and Task B

Overview of deviations	Task A			Task B
	R7	R19	Overall	
Overall deviations (Total)	20	234	254	133
Process deviations (Total)	18	214	232	88
Product deviations (Total)	2	20	22	45
Overall deviations (Avg. / designer)	1.3	14.6	15.9	8.3
Process deviations (Avg. / designer)	1.1	13.4	14.5	5.5
Product deviations (Avg. / designer)	0.1	1.3	1.4	2.8
Deviations per category				
Wrong pattern, wrong parameters	13	61	74	36
Pattern testing/learning; trials	4	79	83	10
Problems with pattern implementation	0	0	0	30
Problems with identification of activities	NA	NA	NA	12
Dead end and dead end resolution	0	57	57	0
Other	1	17	18	0

Most of the designers started the modeling with little problems (i.e., from the 232 process deviations only 18 are related to region R7) that were mostly caused by wrong pattern usage (e.g., loop instead of conditional) or wrong parameter settings (e.g., activity inserted at wrong position). The remaining 214 deviations

occurred in region R19, which only comprises two activities more than R7, but has higher structural complexity. Interestingly, when combining the patterns in an optimal way, the creation of both parts requires a similar number of change patterns (i.e., 7 patterns for R7 and 11 patterns for R19). When analyzing the process deviations related to R19, it turned out that process designers especially faced difficulties in creating region R16. Its nested structure forces designers to apply patterns in a certain ordering (i.e., requires an effective problem solving strategy including look ahead). For example, assuming that R8, R9, and R10 have already been inserted, the insertion of R13 in parallel to the already inserted regions leads to a dead end (i.e., the solution model can only be reached by deleting already created parts). Most designers (11 out of 16) ended up at least once in a dead end when trying to create the solution model. 3 of these 11 designers did not try to resolve the dead end, i.e., R16 is modeled incorrectly in their final models. The remaining designers (8 out of 11) tried to resolve the dead end by backtracking in the modeling process (i.e., 57 deviations). Partially, it took them several trials until they found a problem solving strategy suitable for constructing the respective fragment (i.e., 79 deviations). Strategies for constructing R16 included the usage of dummy activities and experiments to test and learn the functioning of the patterns. In turn, 5 out of 16 designers faced relatively little difficulties with the creation of this fragment since their initial strategy turned out to be effective, i.e., they were able to build the solution model in a straight-forward manner. In addition to problems related to the creation of R16, process deviations were caused in the context of single activities of region R19. Again, the usage of wrong patterns or wrong parameter values was the primary source of deviations (i.e., 61 deviations).

4.3 Results Related to Task B

Regarding Task B, a minimum of 19 change operations is needed to create the correct solution model. Overall, we identified 133 deviations, 88 process deviations and 45 product deviations (cf. Table 1). Process deviations per process designer ranged from 1 to 17 with an average of 5.5 deviations. Product deviations per designer ranged from 0 to 5 deviations with an average of 2.8 deviations. Product deviations were partially caused by ambiguities in the textual description, partially by mismatches between the textual description and the final models and presumably rather stem from problems with the domain than from actual pattern usage. Since subjects did not have a process model given as a template like in Task A, but had to build the model themselves starting from an informal requirements description, it is little surprising that the percentage of product deviations is much higher compared to Task A. Regarding process deviations, for 12 out of 88, there is clear evidence that they stem from problems with the domain rather than from problems caused by pattern usage (i.e., subjects were not sure whether to include certain model parts as activities and presumably lacked schemata for information extraction). For 30 out of 88 process deviations, there are clear indications that they were caused by problems in pattern usage. As detailed later on, all these deviations occurred in the context

of region R4 and are the result of tooling problems, i.e., caused by patterns implementation. Additional 10 deviations seem to stem from problems in pattern usage and were caused by process designers testing the functioning of certain patterns. In turn, the remaining 36 deviations were caused by process designers initially selecting the wrong patterns (e.g., embedding an activity in a loop instead of embedding it in a conditional branch), and by using the correct pattern with incorrect parameters (e.g., inserting the correct activity at a wrong position). Even though most of these deviations rather seem to be domain related, we cannot conclude with certainty from our data whether this indicates problems caused by pattern usage (i.e., mapping the mental model to the available patterns) or problems with the domain (i.e., incorrect mental model).

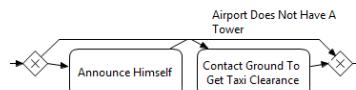


Fig. 4. Problem with Combined Pattern Usage

Overall, subjects faced relatively little problems related to the usage of patterns when working on Task B. In particular, they did not have any notable problem when inserting activities in a sequence, making an activity optional, or inserting an activity in parallel to another one. The only exception was a region with two exclusive branches, which caused significant problems (cf. R4 in Fig. 4). To model this region correctly, process designers had to first insert one of the activities using pattern *Serial Insert*, subsequently use pattern *Embed Process Fragment in Conditional Branch* to make the previously inserted activity optional, then use pattern *Serial Insert* to insert the second activity, and finally use pattern *Update Condition* to insert a transition condition. Even though the subjects started correctly in the construction of this region—problems with the automatic layout—CEP made 5 out of 16 process designers think that the second *Serial Insert* pattern has not been applied correctly resulting into partially long detours (cf. Fig. 4). As a consequence, they tried to apply the pattern several times even though their initial solution would have been correct. 4 of the 5 process designers facing this problem realized after a few trials that the pattern had been applied correctly. Just 1 of 5 process designers, however, created a workaround solution that correctly reflected the requirements, but was a bit more complicated than the optimal solution (i.e., instead of creating one process fragment with two conditional branches, this designer created two process fragments with one optional activity each). Overall 30 out of 88 process deviations were caused by this problem. Interestingly, when faced with the same modeling structure later in the modeling process again (i.e., R7), the process designers did not have these problems anymore, but apparently learned how to use the patterns in combination to model such construct (4 process designers) or how to circumvent the situation with a workaround (1 process designer). While process designers had no problem solving strategy available when constructing R4, they could rely on the plan schemata developed in R4 for the construction of R7.

5 Subjective Perception of Model Creation

This section addresses research question R2 which deals with the subjective perception of process designers using change patterns. We investigate their ease of use and perceived usefulness for creating process models and discuss feedback provided by the participants after the exploratory study.

5.1 Perceived Ease of Use and Perceived Usefulness

To assess in how far process designers with moderate process modeling knowledge consider the CEP change pattern modeler as easy to use and useful, we asked them to fill out the *Perceived Ease of Use* and the *Perceived Usefulness* scales from the Technology Acceptance Model (TAM) [25] after the modeling session. Both scales consist of six 7-point Likert items, ranging from Extremely likely (1) over Neither Likely nor Unlikely (4) to Extremely Unlikely (7). On average, for the Perceived Ease of Use scale the process designer responded with 2.88, which approximately relates to Slightly Likely (3). For the Perceived Usefulness scale, in turn, the process designer in average responded with 3.49, which approximately relates to Slightly Likely (3). Hence, we conclude that process designers find it in average slightly likely that it would be easy to learn and use change patterns.

5.2 Qualitative Feedback Regarding Change Pattern Usage

We additionally asked participants for qualitative feedback with the usage of change patterns. The obtained feedback revealed usability issues, which are in line with the results reported in Sect. 4.2 and 4.3, and which at least partially explain why perceived ease of use and perceived usefulness did not receive better scores. The qualitative evaluation shows that perceived ease of use and perceived usefulness of the patterns heavily depends on process characteristics. Several designers stated that they perceive change patterns especially useful for models that are rather simple, since in this case change patterns allow them to speed up modeling. For more complex models (e.g., highly nested models), process designers rather prefer using change primitives, since this gives them more flexibility: *“I like both ways of working, with patterns and without. After getting a bit used to the patterns I find them easier to not let me make mistakes, for thinking and sketching I find the other approach easier.”* Since the available pattern set was limited, process designers were partially forced to delete big fragments when their problem solving strategy turned out to be ineffective (cf. Sect. 4.2 and 4.3). This was especially true for model regions with complex structure requiring more sophisticated problem solving strategies on how to combine patterns. This is reflected by the following statement of one participant: *“From my point of view, it always depends on whether the process is clear, which makes it easy to use change patterns. However, if the process somehow is really complex and it’s hard to think about everything before starting to model a process it’s better to avoid using change patterns, since once you’ve model something wrong, it could happen that you have to remodel many parts of the process.”* This statement

indicates that limitations of working memory might be a bottleneck, especially when designers lack schemata for efficiently extracting and processing information and problem solving strategies on how to best combine patterns. To combine the strength of patterns-based modeling and the modeling of change primitives, designers expressed the wish for a modeling environment allowing for the combination of both modeling styles: “*If the change pattern functionality would be included into standard modeling tools it would be a very useful addition*”.

6 Discussion and Limitations

This section discusses the results and presents limitations that pose potential threats to their validity.

6.1 Discussion

The results described in Sect. 4 reveal that simple control flow structures without any nesting can be well managed by most designers. Presumably, designers were able to quickly develop plan schemata for simple models. In general, only few problems, which can be directly attributed to pattern usage, could be observed for model parts without any nesting. Even when faced with the modeling environment for the first time, subjects did not have any notable problems when inserting activities in sequences, making an activity optional, or inserting an activity in parallel. There was only one exception where detours in the modeling process were caused by a poor implementation of the graphical layout of the CEP modeler, which will be addressed in a future version of this software to improve perceived ease of use. Faced with more complex control flow structures, in turn, the structural restrictions imposed by modeling based on change patterns led to considerable problems with model construction partially resulting into long detours or incorrect models. These findings are underlined by feedback of the participants who appreciate the correctness-by-construction guarantees, but feel restricted when faced with complex control flow constructs (cf. Sect. 5). Clearly, designers could not rely on existing plan schemata for such complex structures, forcing designers to apply means-ends analysis for solving the problem.

Difficulties faced by process designers can partially be explained by the available patterns set. Even though the patterns available to process designers cover all basic control-flow patterns (i.e., sequence, exclusive/parallel branchings, and loops), the pattern set we used turned out to be insufficient for efficient model construction. Especially in the context of Task A most process designers had to delete parts of their model due to dead ends when trying to construct region R16. Having a pattern *Move Process Fragment* [4] available would presumably address many of the challenges faced by process designers and facilitate resolution of dead ends. Therefore, we plan to extend CEP and to conduct another study to test whether this will lead to the expected benefits and improve perceived ease of use and usefulness of change patterns modeling.

6.2 Limitations

As every research, this work is subject to limitations. The fact that the sample size (16 participants) was relatively small certainly constitutes a threat regarding the generalization of our results. In addition, using students instead of professionals poses another validity threat. However, we are mildly optimistic about the usefulness of the presented insights on the basis of modeling behavior of graduate students, since [30] identified that such subjects perform equally well in process modeling tasks as some professional designers. However, we acknowledge that process designers experienced with the usage of change patterns will presumably face less problems during model creation. Another limitation relates to the fact that we used only two different modeling tasks (with different complexity) in our study. The analysis indicated that difficulties during model creation strongly depend on model characteristics. It is questionable in how far results may be generalized to models with different characteristics. As a consequence, we plan additional experiments testing the influence of model structure on difficulties in change pattern usage. For some of the deviations in the context of Task B we cannot conclude with certainty from our data whether the problems were caused by pattern usage or insufficient domain knowledge. To single out these factors we will conduct further studies with a setup as suggested in [8]. Regarding the internal validity, to alleviate the threat related to the classification of reasons for deviations, a consensus-building process [29] was performed by two authors of the paper.

7 Related Work

Our work is closely related to research on the PPM and process model creation patterns. Research on the process of modeling typically deals with interaction of different parties focusing on structured discussions among system analysts and domain experts [31,22]. The procedure of developing process models in a team is analyzed in [32] and characterized as negotiation process. Participative modeling is discussed in [33]. Each of these works builds on observations of modeling practice and distills normative procedures for steering the process of modeling toward successful completion. Hereby, the focus is on the effective interaction between the involved stakeholders. Our work is complimentary to this perspective through its focus on the *formalization* of the process model. The interactions with the modeling environment have been investigated in [11], identifying three distinct modeling styles. In turn, [10] demonstrates that a structured modeling style leads to models of better quality, and [11,34] suggest different visualization techniques for obtaining an overview of the PPM. [35] investigates the PPM using eye movement analysis. Common to all these works is the focus on interactions with the modeling environment using change primitives, while this paper investigates the use of change patterns. Also related to our work is the usage of change patterns for process schema creation. For example, AristaFlow allows modeling a sound process schema based on an extensible set of change patterns [12]. In turn, [36] describes a set of pattern compounds, similar to change

patterns, allowing for the context-sensitive selection and composition of workflow patterns during process modeling. Complementary to existing research on process model creation based on patterns, which has a strong design focus, this paper provides first empirical insights into the usage of change patterns.

8 Summary

While work related to the PPM has emerged as a new stream of research in recent years, little is known about this process when utilizing change patterns. In this exploratory study we investigate the challenges, process designers are facing when creating process models based on change patterns as well as their subjective perception regarding the usage of these patterns. Our results show that process designers face relatively little difficulties when creating simple control-flow structures. When faced with more complex process structures, the structural restrictions imposed by change patterns caused considerable problems for most of the process designers. Building respective structures efficiently (i.e., without detours) requires process designers to look ahead, since patterns cannot always be arbitrarily combined. This need for looking ahead is a fundamental difference compared to process model creation using change primitives and did not only lead to observable difficulties, but was also perceived as challenging and restrictive by subjects. The exploratory study not only confirmed that the creation of process models using change patterns impacts the PPM, but also gives advice regarding the improvement of the modeling approach based on change patterns. In particular, it showed that a basic set of change pattern as used for the exploratory study is not sufficient for efficient model creation.

References

1. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of Business Process Modeling. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) *Business Process Management*. LNCS, vol. 1806, pp. 30–49. Springer, Heidelberg (2000)
2. Kock, N., Verville, J., Danesh-Pajou, A., DeLuca, D.: Communication flow orientation in business process modeling and its effect on redesign success: Results from a field study. *Decision Support Systems* 46, 562–575 (2009)
3. Mendling, J., Verbeek, H.M.W., van Dongen, B.F., van der Aalst, W.M.P., Neumann, G.: Detection and prediction of errors in EPCs of the SAP reference model. *Data and Knowledge Engineering* 64, 312–329 (2008)
4. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. *Data and Knowledge Engineering* 66, 438–466 (2008)
5. Rinderle-Ma, S., Reichert, M., Weber, B.: On the formal semantics of change patterns in process-aware information systems. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) *ER 2008*. LNCS, vol. 5231, pp. 279–293. Springer, Heidelberg (2008)
6. Reichert, M., Dadam, P.: ADEPTflex: Supporting Dynamic Changes of Workflow without Losing Control. *JIIS* 10, 93–129 (1998)

7. Casati, F.: Models, Semantics, and Formal Methods for the design of Workflows and their Exceptions. PhD thesis, Milano (1998)
8. Soffer, P., Kaner, M., Wand, Y.: Towards Understanding the Process of Process Modeling: Theoretical and Empirical Considerations. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011 Workshops, Part I. LNBP, vol. 99, pp. 357–369. Springer, Heidelberg (2012)
9. Pinggera, J., Zugal, S., Weidlich, M., Fahland, D., Weber, B., Mendling, J., Reijers, H.A.: Tracing the Process of Process Modeling with Modeling Phase Diagrams. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011 Workshops, Part I. LNBP, vol. 99, pp. 370–382. Springer, Heidelberg (2012)
10. Claes, J., et al.: Tying Process Model Quality to the Modeling Process: The Impact of Structuring, Movement, and Speed. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 33–48. Springer, Heidelberg (2012)
11. Pinggera, J., Soffer, P., Zugal, S., Weber, B., Weidlich, M., Fahland, D., Reijers, H.A., Mendling, J.: Modeling Styles in Business Process Modeling. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (eds.) BPMDS 2012 and EMMSAD 2012. LNBP, vol. 113, pp. 151–166. Springer, Heidelberg (2012)
12. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Comp. Scie. - R&D 23, 81–97 (2009)
13. Gray, P.: Psychology. Worth Publishers (2007)
14. Tracz, W.: Computer programming and the human thought process. Software: Practice and Experience 9, 127–137 (1979)
15. Jeffries, R., Turner, A., Polson, P., Atwood, M.: The Process Involved in Designing Software. In: Cognitive Skills and Their Acquisition, pp. 255–283. Erlbaum (1981)
16. Rist, R.: Schema Creation in Programming. Cognitive Science 13, 389–414 (1989)
17. Kant, E., Newell, A.: Problem Solving Techniques for the design of algorithms. Information Processing & Management 20, 97–118 (1984)
18. Anderson, J.: Acquisition of cognitive skill. Psychological Review 89, 369–406 (1982)
19. Guindon, R., Curtis, B.: Control of cognitive processes during software design: what tools are needed? In: Proc. CHI 1988, pp. 263–268 (1988)
20. Sweller, J.: Cognitive load during problem solving: Effects on learning. Cognitive Science, 257–285 (1988)
21. Brooks, R.: Towards a theory of the cognitive processes in computer programming. International Journal of Man-Machine Studies 9, 737–751 (1977)
22. Hoppenbrouwers, S.J.B.A., (Erik) Proper, H.A., van der Weide, T.P.: A fundamental view on the process of conceptual modeling. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 128–143. Springer, Heidelberg (2005)
23. Petre, M.: Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. Commun. ACM, 33–44 (1995)
24. Pinggera, J., Zugal, S., Weber, B.: Investigating the Process of Process Modeling with Cheetah Experimental Platform. In: Proc. ER-POIS 2010, pp. 13–18 (2010)
25. Davis, F.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly 13, 319–340 (1989)
26. Pinggera, J., Zugal, S., Weber, B., Fahland, D., Weidlich, M., Mendling, J., Reijers, H.A.: How the Structuring of Domain Knowledge Can Help Casual Process Modelers. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 445–451. Springer, Heidelberg (2010)

27. Fahland, D., Woith, H.: Towards process models for disaster response. In: Proc. PM4HDPS 2008, pp. 254–265 (2008)
28. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer (2012)
29. Recker, J., Safrudin, N., Rosemann, M.: How novices design business processes. *Inf. Syst.* 37, 557–573 (2012)
30. Reijers, H., Mendling, J.: A study into the factors that influence the understandability of business process models. *IEEE Transactions on Systems Man and Cybernetics, Part A* 41, 449–462 (2011)
31. Frederiks, P., Weide, T.: Information modeling: The process and the required competencies of its participants. *Data and Knowledge Engineering* 58, 4–20 (2006)
32. Rittgen, P.: Negotiating Models. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 561–573. Springer, Heidelberg (2007)
33. Stirna, J., Persson, A., Sandkuhl, K.: Participative Enterprise Modeling: Experiences and Recommendations. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 546–560. Springer, Heidelberg (2007)
34. Claes, J., Vanderfeesten, I., Pinggera, J., Reijers, H., Weber, B., Poels, G.: Visualizing the Process of Process Modeling with PPMCharts. In: Proc. TAPoViz 2012, pp. 744–755 (2013)
35. Pinggera, J., Furtner, M., Martini, M., Sachse, P., Reiter, K., Zugal, S., Weber, B.: Investigating the Process of Process Modeling with Eye Movement Analysis. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBP, vol. 132, pp. 438–450. Springer, Heidelberg (2013)
36. Gschwind, T., Koehler, J., Wong, J.: Applying patterns during business process modeling. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 4–19. Springer, Heidelberg (2008)

Synthesizing a Library of Process Templates through Partial-Order Planning Algorithms

Andrea Marrella¹ and Yves Lespérance²

¹ Sapienza - Universitá di Roma, Italy

² York University, Toronto

Abstract. The design time specification of dynamic processes can be time-consuming and error-prone, due to the high number of tasks involved and their context-dependent nature. Such processes frequently suffer from potential interference among their constituents, since resources are usually shared by the process participants and it is difficult to foresee all the potential tasks interactions in advance. Concurrent tasks may not be independent from each other (e.g., they could operate on the same data at the same time), resulting in incorrect outcomes. To address these issues, we propose an approach that exploits partial-order planning algorithms for automatically synthesizing a library of process template definitions for different contextual cases. The resulting templates guarantee sound concurrency in the execution of their activities and are reusable in a variety of partially-known contextual environments.

1 Introduction

Current workflow technology is based on the idea that there *always* exists an underlying fixed process that can be used to automate the work [1]. Once identified, a process is formalized into a *process model* which captures every possible case (i.e., *process instance*) to be executed at run-time through a Process Management System (PMS). This approach works for processes where procedures are well known, repeatable and can be planned in advance with some level of detail. In recent years, the need to deal with *dynamic processes* and provide support for flexible process management has emerged as a leading research topic in the Business Process Management (BPM) domain [2]. In a dynamic process, the sequence of tasks depends heavily on the specifics of the context (e.g., which resources are available and what particular options exist at the time), and it is often unpredictable how the process will unfold. The design-time specification of all possible cases requires an extensive manual effort for the process designer, who has to anticipate all potential alternatives into the process model, in an attempt to deal with the context dependent nature of these processes (cf. Section 2). Such processes do not have the same level of repeatability of classical business processes, and the execution changes on a case-by-case basis, generating instances that are different almost every time, depending on the context.

In this paper, we present an approach that allows us to automatically synthesize a *library of process templates* starting from a representation of the contextual

domain in which the process is embedded in and from an extensive repertoire of tasks defined for such a context. A template depicts the best-practice procedure drawn up with whatever contextual information available at the time; it describes a recommended control flow for the process that can be enacted in a range of states satisfying the context conditions. In order to build process templates, we make use of *partial-order planning algorithms* (aka POP [3]), which guarantee some interesting properties in the construction of the template:

- *Sound concurrency.* A template has the property of *sound concurrency* in the execution of its concurrent activities, that are proven to be effectively *independent* one from another (i.e., at runtime there is no risk of interference between concurrent tasks, since they cannot affect the same data).
- *Executability in partially known environments.* Once synthesized, a template can be executed in several starting states, since it (usually) requires a fragment of the knowledge of the starting state to successfully achieve its objectives. We identify the *weakest preconditions* of process templates, and all the states satisfying such preconditions are good candidates for executing them.

We exploit the idea behind POP of representing flexible plans that enables deferring decisions. Instead of committing prematurely to a complete, totally ordered sequence of actions, plans are represented as a partially ordered set, and only the required ordering decisions are recorded. A process template is generated on the basis of such a set of activities, and we are able to identify what knowledge about the starting state is required for successful template execution. Moreover, we build step-by-step a library of process template specifications and support efficient retrieval of appropriate templates in partially known environments.

2 A Running Example

Let us consider the emergency management scenario described in Fig. 1(a). It concerns a train derailment and depicts a map of the area (as a 4x4 grid of locations) where the disaster happened. We suppose that the train is composed of a locomotive (located in *loc33*) and two coaches (located in *loc32* and *loc31* respectively). The goal of an incident response plan defined for such a context is to evacuate people from the coach located in *loc32*, to extinguish a fire in the coach in *loc31* and finally to take pictures for evaluating possible damages to the locomotive, located in *loc33*. Thus, a response team can be sent to the derailment scene. The team is composed of four first responders (in the remainder, we refer to them as *actors*) and two robots, initially located in *loc00*. We assume that actors are equipped with mobile devices (for picking up and executing tasks) and provide specific capabilities. For example, actor *act1* is able to extinguish fires, while *act2* and *act3* can evacuate people from train coaches. The two robots, instead, may take pictures and remove debris from specific locations. Each robot has a battery and each action consumes a given amount of battery charge. When the battery of a robot is discharged, actor *act4* can charge it. Fig. 1(b) summarizes the above.

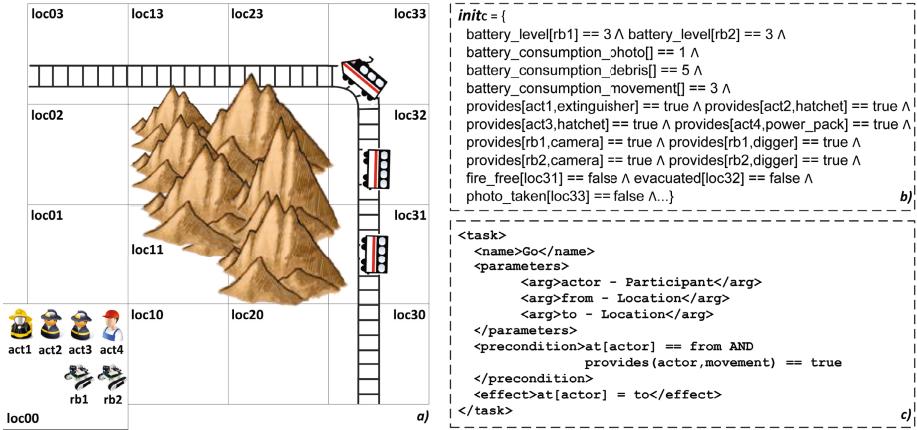


Fig. 1. Area and context of the intervention

The definition of an incident response plan as a business process involves a dynamically selected set of activities to be executed on the field by the first responders. Since the process may be different every time it is defined because it strictly depends on the actual contextual information (the positions of actors/robots, the battery level of robots, etc.), it is unrealistic to assume that the process designer can pre-define all the possible process models for dealing with this environment (apparently simple). Moreover, if contextual data describing the environment are known, the synthesis of a process dealing with such an environment is not straightforward, as the correctness of the process model is constrained by the values (or combination of values) of contextual data. A simple approach to solving our problem is to build a process as a sequence of activities, e.g., the sequence of actions shown in Fig. 2. However, this solution is highly “inefficient”, as many actions are independent, and they could be executed concurrently to reduce intervention time; e.g., a robot could take pictures in parallel with the extinguishing of the fire in *loc31*. But, at the same time, a process designer may find it difficult to organize activities for concurrent execution, since each action, for its executability, depends on the values of contextual data (e.g., a robot needs enough battery charge for moving into a location and taking pictures or removing debris). Also dependencies between actions play a key role in the definition of the process model (e.g., in order to evacuate people at *loc32*, a robot must have removed the debris beforehand). Finally, a process designer tends to represent more contextual information than that strictly needed for defining a process. For example, the process in Fig. 2 does not involve actor *act3*, meaning that any information concerning *act3* (e.g., its capabilities, its location, etc.) is not required for synthesizing and executing the process. To overcome the above issues, we propose a solution that involves exploiting *partial-order planning* for generating a library of *process templates* for different contextual cases. Our templates provide *sound concurrency* in the execution of their activities and are executable in *partially known environments*.

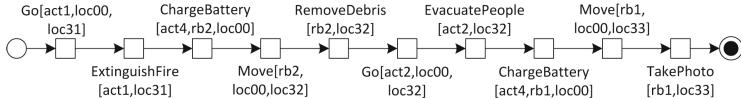


Fig. 2. A process dealing with the scenario of Fig. 1

3 Partial-Order Planning

Planning systems are problem-solving algorithms that operate on explicit representations of states and actions. The standard representation language for classical planners is known as the Planning Domain Definition Language (PDDL [4]); it allows us to formulate a problem PR through a set of possible actions, the description of the initial state of the world $init_{PR}$ and of the desired goal condition $goal_{PR}$. The set of all action definitions Ω is the *domain* PD of the planning problem. Each action $a \in \Omega$ has a preconditions list (stating the atomic conditions under which an action can be executed) and an effects list to be applied on the state of the world, denoted respectively as Pre_a and Eff_a . A planner that works on such inputs generates a sequence of actions (the *plan*) that corresponds to a path from the initial state to a state meeting the goal condition.

In this paper, we focus on *Partial-Order Planning (POP)* [3], a specific kind of plan-space search algorithm. A plan space is an implicit directed graph whose nodes are *partially specified plans* and whose edges correspond to refinement operations that further complete a partial plan, i.e., to achieve an open goal or to remove possible inconsistencies. POP takes as input a PDDL planning problem and searches the space of partial plans without committing to a totally ordered sequence of actions. Basically, a **partial plan** is a tuple $P = (A, O, CL)$, where $A \subseteq \Omega$ is a set of (ground) actions, O is a set of *ordering constraints* over A , and CL is a set of *causal links* over A . Ordering constraints O are of the form $a \prec b$, which is read as “ a before b ” and means that action a must be executed sometime before action b , but not necessarily immediately before. Causal links CL may be represented as $c \xrightarrow{p} d$, which is read as “ c achieves p for d ” and means that p is an effect of action c and a precondition for action d . A precondition without a causal link requires further refinement to the plan to establish it, and is considered to be an *open condition* in the partial plan. A classical POP algorithm starts with a null partial plan P and keeps refining it until a solution plan is found. The null partial plan contains two dummy actions $a_0 \prec a_\infty$ where the preconditions of a_∞ correspond to the top level goals $goal_{PR}$ of the problem, and the effects of a_0 correspond to the conditions in $init_{PR}$. Intuitively, a refinement operation avoids adding to the partial plan any constraints that are not strictly needed for addressing the refinement objective. This is called the *least commitment principle* [3], and its advantage is that decisions about action ordering are postponed until a decision is forced; constraints are not added to a partial plan unless strictly needed, thus guaranteeing flexibility in the execution of the plan and by allowing actions to run concurrently. A **consistent** plan is defined as a plan with no cycles in the ordering constraints and no conflicts with the causal links. A consistent plan with no open conditions is a **solution** [3].

4 Process Templates

The synthesis of a dynamic process requires a tight integration of process activities and contextual data in which the process is embedded in. The context is represented in the form of a *Domain Theory* D , that captures a set of tasks $t_i \in \mathsf{T}$ (with $i \in 1..n$) and supporting information, such as the people/agents that may be involved in performing the process (roles or participants), the data and so forth. Tasks are collected in a specific repository, and each task can be considered as a single step that consumes input data and produces output data. Data are represented through some ground atomic terms $v_1[y_1], v_2[y_2], \dots, v_m[y_m] \in \mathsf{V}$ that range over a set of tuples (i.e., unordered sets of zero or more attributes) y_1, y_2, \dots, y_m of *data objects*, defined over some *data types*. In short, a data object depicts an entity of interest; for example, in our scenario we need to define data objects for representing participants (e.g., data type $\text{Participant} = \{\text{act1}, \text{act2}, \text{act3}, \text{act4}, \text{rb1}, \text{rb2}\}$), capabilities (e.g., data type $\text{Capability} = \{\text{extinguisher}, \text{movement}, \dots, \text{hatchet}\}$) and locations in the area (e.g., data type $\text{Location} = \{\text{loc00}, \text{loc10}, \dots, \text{loc33}\}$). Each tuple y_j may contain one or more data objects belonging to different data types. The domain $\text{dom}(v_j[y_j])$ over which a term is interpreted can be of various types: (i) *Boolean*: $\text{dom}(v_j[y_j]) = \{\text{true}, \text{false}\}$, (ii) *Integer*: $\text{dom}(v_j[y_j]) = \mathbb{Z}$, (iii) *Functional*: the domain contains a fixed number of data objects of a designated type. Terms can be used to express properties of domain objects (and relations over objects). In our example, we may need boolean terms for expressing the presence of a fire in a location (e.g., $\text{fire-free}[loc : \text{Location}] = (\text{bool} : \text{Boolean})$), integer terms for representing the battery charge level of each robot (e.g., $\text{batteryLevel}[prt : \text{Participant}] \in \mathbb{Z}$) or functional terms for recording the position of each actor in the area (e.g., $\text{at}[prt : \text{Participant}] = (loc : \text{Location})$). Moreover, since each task has to be assigned to a participant that provides all of the skills required for executing that task, there is the need to consider the participants “capabilities”. This can be done through a boolean term $\text{provides}[prt : \text{Participant}, cap : \text{Capability}]$ that is *true* if the capability cap is provided by prt and *false* otherwise.

Each task is annotated with *preconditions* and *effects*. Preconditions can be used to constrain the task assignment and must be satisfied before the task is applied, while effects establish the outcome of a task after its execution. Note that, as shown in Fig. 3(a), our approach treats each task as a “black box” and no assumption is made about its internal behavior (we consider the task execution as an instantaneous activity).

Definition 1. A task $t[x] \in \mathsf{T}$ consists of:

- a tuple of data objects x as input parameters;
- a set of preconditions $Pret_t$, represented as the conjunction of k atomic conditions defined over some specific terms, $Pret_t = \bigwedge_{l \in 1..k} pret_{t_l}$. Each $pret_{t_l}$ can be represented as $\{v_j[y_j] \text{ op } expr\}$, where:
 - $v_j[y_j] \in \mathsf{V}$ is an atomic term, with $y_j \subseteq x$, i.e., admissible data objects for y_j need to be defined as task input parameters;

- An **expr** can be a boolean value (if v_j is a boolean term); an input parameter identified by a data object (if v_j is a functional term); an integer number or an expression involving integer numbers and/or terms, combined with the arithmetic operators $\{+, -\}$ (if v_j is a integer term);
- $op \in \{<, >, ==, \leq, \geq\}$ is a relational operator.
- a set of deterministic effects Eff_t , represented as the conjunction of h atomic conditions defined over some specific terms, $Eff_t = \bigwedge_{l \in 1..h} eff_{t_l}$. Each eff_{t_l} (with $l \in 1..h$) can be represented as $\{v_j[y_j] \ op \ expr\}$, where:
 - $v_j[y_j] \in V$ and **expr** are defined as for preconditions.
 - $op \in \{=, +, -, =\}$ is used for assigning ($=$) to a term a value consistent with the **expr** field or for incrementing ($+ =$) or decrementing ($- =$) an integer term by that value.

For example, the task *Go* described in Fig. 1(c) involves two parameters *from* and *to* of type *Location* and a parameter *actor* of type *Participant*. An instance of *Go* can be executed only if *actor* is currently at the starting location *from* and provides the required capabilities for executing the task. As a consequence of task execution, the actor moves from the starting to the arrival location, and this is reflected by assigning to the term *at[actor]* the value *to* in the effect.

Modeling a business process involves representing how a business pursues its objectives/goals. The goal may vary depending on the specific *Process Case C* to be handled. A case C reflects an instantiation of the domain theory D with a starting condition $init_C$ and a goal condition $goal_C$. Both conditions are conjunctions of atomic terms. We do not assume complete information about $init_C$; this means we allow a process designer to instantiate only the atomic terms necessary for representing what is known about the starting state, i.e., $init_C = \{v_1[y_1] == val_1 \wedge \dots \wedge v_j[y_j] == val_j\}$, where val_j (with $j \in 1..m$) represents the j-th value assigned to the j-th atomic term. Fig. 1(b) shows a portion of $init_C$ concerning the scenario depicted in Fig. 1(a). The goal is a condition represented as a conjunction of some specific terms we want to make true through the execution of the process. For example, in the scenario shown in Section 2, the goal has to be represented as : $goal_C = \{fire_free[loc31] == true \wedge evacuated[loc32] == true \wedge photo_taken[loc33] == true\}$. The syntax of goal conditions is the same as for tasks preconditions.

A state is a complete assignment of values to atomic terms in V . Given a case C, an intermediate state $state_{C_i}$ is the result of i tasks performed so far, and atomic terms in V may be thought of as “properties” of the world whose values may vary across states.

Definition 2. A task t can be performed in a given $state_{C_i}$ (and in this case we say that t is **executable** in $state_{C_i}$) iff $state_{C_i} \vdash Pre_t$, i.e. $state_{C_i}$ **satisfies** the preconditions Pre_t for the task t .

Moreover, if executed, the effects Eff_t of t modify some atomic terms in V and change $state_{C_i}$ into a new state $state_{C_{i+1}} = update(state_{C_i}, Eff_t)$. The *update* function returns the new state obtained by applying effects Eff_t on the current state $state_{C_i}$. Starting from a domain theory D, a *Process Template*

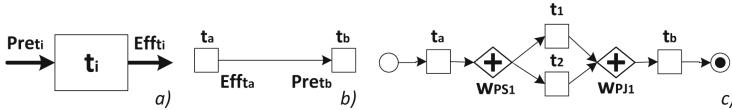


Fig. 3. Task anatomy (a), causality (b) and concurrency (c) in a process model

captures a partially ordered set of tasks, whose successful execution (i.e., without exceptions) leads from $init_C$ to $goal_C$. Formally, we define a template as a directed graph consisting of tasks, gateways, events and transitions between them.

Definition 3. Given a domain theory D , a set of tasks T and a case C , a Process Template PT is a tuple (N, L) where:

- $N = T \cup E \cup W$ is a finite set of nodes, such that :
 - T is a set of tasks instances, i.e., occurrences of a specific task $t \in T$ in the range of the process template;
 - E is a finite set of events, that consists of a single start event \circlearrowleft and a single end event \circlearrowright ;
 - $W = W_{PS} \cup W_{PJ}$ is a finite set of parallel gateways, represented in the control flow with the \diamond shape with a “plus” marker inside.
- $L = L_T \cup L_E \cup L_{W_{PS}} \cup L_{W_{PJ}}$ is a finite set of transitions connecting events, task instances and gateways:
 - $L_T : T \rightarrow (T \cup W_{PS} \cup W_{PJ} \cup \circlearrowleft)$
 - $L_E : \circlearrowleft \rightarrow (T \cup W_{PS} \cup \circlearrowright)$
 - $L_{W_{PS}} : W_{PS} \rightarrow 2^T$
 - $L_{W_{PJ}} : W_{PJ} \rightarrow (T \cup W_{PS} \cup \circlearrowright)$

The constructs used for defining a template are essentially a subset of the BPMN notation [5], a graphical language designed to specify a process in a standardized way. Intuitively, an execution of the process starts at \circlearrowleft and ends at \circlearrowright ; a *task* is an atomic activity executed by the process; *parallel splits* W_{PS} open parallel parts of the process, whereas *parallel joins* W_{PJ} re-unite parallel branches. *Transitions* are binary relations describing in which order the flow objects (tasks, events and gateways) have to be performed, and determine the *control flow* of the template. For example, in Fig. 3(b) we have a relation of *causality* between tasks t_a and t_b , stating that t_a must take place before t_b happens as t_a achieves some of t_b 's preconditions.

An important feature provided by a process template is *concurrency*, i.e., several tasks can occur concurrently. In Fig. 3(c) an example of concurrency between t_1 and t_2 is shown. In order to represent two or more concurrent tasks in a template, the process designer makes use of the parallel gateways, that indicate points of the template in which tasks can be carried out concurrently. A *linearization* of a process template is any linear ordering of the tasks that is consistent with the ordering constraints of the template itself [6]; i.e., a linearization of a partial order is a potential *execution path* of the template from the start event \circlearrowleft to the end event \circlearrowright . For example, the template in Fig. 3(c) has two possible execution paths $r_1 = [\circlearrowleft; t_a; t_1; t_2; t_b; \circlearrowright]$ and $r_2 = [\circlearrowleft; t_a; t_2; t_1; t_b; \circlearrowright]$.

Definition 4. Given a process template PT and an initial state $state_{C_0} \vdash init_C$, a state $state_{C_i}$ is said to be **reachable** with respect to PT iff there exists an

execution path $r = [\bigcirc; t_1; t_2; \dots; t_k; \bigcirc]$ of PT and a task t_i (with $i \in 1..k$) such that $\text{state}_{\mathcal{C}_i} = \text{update}(\text{update}(\dots \text{update}(\text{state}_{\mathcal{C}_0}, \text{Eff}_{t_1}) \dots, \text{Eff}_{t_{i-1}}), \text{Eff}_{t_i})$.

Definition 5. *A task t_1 **affects** the execution of a task t_2 ($t_1 \triangleright t_2$) iff there exists a reachable state $\text{state}_{\mathcal{C}_i}$ of PT (for some initial state $\text{state}_{\mathcal{C}_0}$) such that:*

- (i) $\text{state}_{\mathcal{C}_i} \vdash \text{Pre}_{t_2}$
- (ii) $\text{update}(\text{state}_{\mathcal{C}_i}, \text{Eff}_{t_1}) \not\vdash \text{Pre}_{t_2}$

This means that Eff_{t_1} modify some terms in \mathcal{V} that are required as preconditions for making t_2 executable in $\text{state}_{\mathcal{C}_i}$.

Definition 6. *Given a process template PT , a case \mathcal{C} and an initial state $\text{state}_{\mathcal{C}_0} \vdash \text{init}_{\mathcal{C}}$, an execution path $r = [\bigcirc; t_1; t_2; \dots; t_k; \bigcirc]$ (where $k = |\mathcal{T}|$) of PT is said to be **executable** in \mathcal{C} iff:*

- (i) $\text{state}_{\mathcal{C}_0} \vdash \text{Pre}_{t_1}$
- (ii) for $1 \leq i \leq k-1$, $\text{update}(\text{state}_{\mathcal{C}_{i-1}}, \text{Eff}_{t_i}) \vdash \text{Pre}_{t_{i+1}}$
- (iii) $\text{update}(\text{state}_{\mathcal{C}_{k-1}}, \text{Eff}_{t_k}) = \text{state}_{\mathcal{C}_k} \vdash \text{goal}_{\mathcal{C}}$

Definition 7. *A process template PT is said to be **executable** in a case \mathcal{C} iff any execution path of PT is executable in \mathcal{C} .*

Definition 8. *Given a process template PT , a task t_x is said to be **concurrent** with a task t_z iff there exist two execution paths r_1 and r_2 of PT such that $r_1 = [\bigcirc; t_1; t_2; \dots; t_x; \dots; t_z; \dots; \bigcirc]$ and $r_2 = [\bigcirc; t_1; t_2; \dots; t_z; \dots; t_x; \dots; \bigcirc]$.*

Definition 9. *Two concurrent tasks t_1 and t_2 are said to be **independent** ($t_1 \parallel t_2$) iff $t_1 \not\triangleright t_2$ and $t_2 \not\triangleright t_1$; that is, t_1 does not affect t_2 and vice versa.*

5 On Synthesizing a Library of Process Templates

Our approach is focussed on the development and use of a *library of process templates*. These are reusable processes that achieve specified goals of interest in any starting state that satisfies the template's required preconditions. Specifically, we focus on the use of a POP-based tool that can synthesize complex concurrent process models, while ensuring that concurrent tasks never interfere. The process designer's role is to specify the domain and context in which the template may be executed. Our POP-based tool can then be used to synthesize some candidate process models for the template. If the tool fails to generate a process model or the generated processes are of insufficient quality (e.g., they are too time consuming, unreliable, or lack concurrency), the designer can refine the domain theory and case to obtain better solutions. Once a satisfactory template has been obtained, it is added to the library. The POP-based tool automatically identifies the required preconditions for the template to achieve its goal, meaning the template can be reused whenever a case that matches the template's preconditions arises. The designer maintains the template library over time, in order to have templates that handle effectively most the cases that arise.

The General Framework. Our approach to the definition of a process template (cf. Fig. 4) requires a fundamental shift in how one thinks about

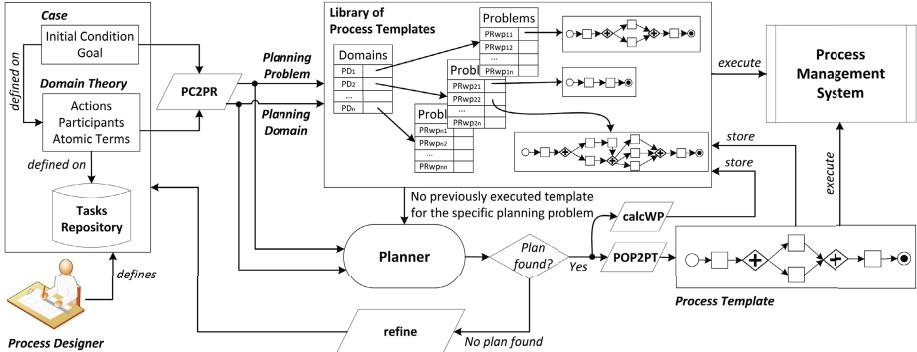


Fig. 4. Overview of the general approach

modeling business processes. Instead of defining a process model “by hand”, the process designer has to address her/his efforts to specifying the Domain Theory D and the Case C to be handled. In particular, s/he has to “guess” the starting condition $init_C$, by instantiating only those atomic terms needed for depicting the context s/he has in mind. This means that $init_C$ can be partially specified, i.e., not all terms need to be instantiated with some value.

Example. Let us consider the scenario depicted in Section 2, represented with a Domain Theory D_1 and a goal condition $goal_{C_1} = \{fire_free[loc31]==true \wedge evacuated[loc32]==true \wedge photo_taken[loc33]==true\}$. Since the process designer may be interested in an emergency process that involves the fewest participants, s/he can start by modeling a starting condition $init_{C_1}$ with information involving only actors $act1$ and $act2$ and the robot $rb1$, while terms involving $act3$, $act4$ and $rb2$ are not explicitly instantiated in $init_{C_1}$.

A specific module named PC2PR is in charge of converting the Domain Theory D and the Case C just defined into the corresponding Planning Domain PD and Planning Problem PR specified in PDDL version 2.1¹ (cf. [4]). Basically, PC2PR implements a function $f_{PC2PR} : (D, init_C, goal_C) \rightarrow (PD, init_{PR}, goal_{PR})$. Since the use of classical partial-order algorithms for synthesizing the template requires the initial state of PR to be a complete state, we make the closed world assumption [7] and assume that every atomic term $v_j[y_j]$ that is not explicitly specified in $init_C$ is assumed to be false (if $v_j[y_j]$ is a boolean term) or “not assigned” (if $v_j[y_j]$ is a integer or a functional term) in $init_{PR}$. At the heart of our approach lies a library of process templates built for specific planning domains and problems/cases. If library templates exist for the current values of PD and PR, we can retrieve an appropriate template and allow to execute it through an external PMS. However, if no template exists for the current values of PD and PR, we can invoke an external POP planner on these same inputs. The planner will try to synthesize a plan fulfilling the goal condition $goal_{PR}$. If the

¹ PDDL 2.1 enables the representation of realistic planning domains, which include operators with universally quantified effects and numeric fluents. However, our formalism does not currently handle conditional effects nor negative preconditions.

planner is unable to find a plan, this suggests there are some missing elements in the definition of the Domain Theory D or in the Case C. Hence, to address this particular case, one can try to *refine* the case C and add information so that it becomes possible to generate a plan. There are many ways to strengthen a problem description, such as adding to the starting condition $init_C$ some terms initially ignored (e.g., to specify the position of every participant), or adding new objects in D or new activities in T (e.g., if a task for extinguish fire is missing). Our approach assumes that one specifies the context step-by-step, and requires the process designer to contribute to the system. **Example.** If the planner is invoked with $init_{PR_1}$ (devised by applying f_{PC2PR} on the triple $D_1, init_{C_1}, goal_{C_1}$), it will not be able to find any plan for the specific problem. This is because $rb1$ does not have enough battery charge for moving, taking pictures and removing debris. The designer can try to add new information to the problem description by instantiating in $init_{C_1}$ all those atomic terms related to actor $act4$, the only one able to charge robot batteries, and devises a new starting condition $init_{C_2}$ (and, consequently, a new initial planning state $init_{PR_2}$). A planner invoked with $init_{PR_2}$ is finally able to find a consistent plan P_1 satisfying $goal_{PR_1}$.

When the POP planner is able to find a partially ordered plan P consistent with the actual contextual information, three further steps are required. First we need to translate the plan into a template PT that preserves the ordering constraints imposed by the plan. A **solution plan** is a three-tuple $P = (A, O, CL)$ that specifies the causal relationships for the actions $a_i \in A$, but without specifying an exact order for executing them. Since the actions and the set of ordering constraints must be represented explicitly as nodes and transitions in the template, we developed a module POP2PT implementing a function $f_{POP2PT} : P \rightarrow PT$ that takes as input P and converts it into a template PT. It works by first finding the immediate predecessors/successors of actions in the plan using the ordering constraints, and then constructing the desired plan template, inserting parallel splits (resp. join) gateways when an action has more than one immediate successor (resp. predecessor). **Example.** By applying f_{POP2PT} to P_1 , we devise the template PT_1 in Fig. 5(a). Dashed arrows are causal links that imply an ordering constraint between pairs of tasks. For example, the ordering constraint between $Go[act1, loc00, loc31]$ and $ExtinguishFire[act1, loc31]$ is derived from the fact that Go has the effect $at[act1]=loc31$ that is needed by $ExtinguishFire$ as precondition (i.e., $act1$ has to be located in $loc31$ for extinguish the fire in that location).

Secondly, our approach *infers* the weakest preconditions w_{PT} about the starting state that are required for the template to achieve its goal. The module we use for inferring w_{PT} is called calcWP and works by analyzing the set of causal links CL computed by the POP planner, to see which logical facts f_k are involved in causal links that originate from the dummy start action a_0 and end in some $a_k \in A$. More formally:

$$\forall (cl_k, f_k, a_k) \text{ s.t. } cl_k = (a_0 \xrightarrow{f_k} a_k) \in CL, \text{ then } f_k \in w_{PT}. \quad (1)$$

Observe that the effects of $a_0 \in A$ specify all atomic facts that are true in the starting state $init_{PR}$. The initial facts that are actually required for the plan to be

executable and achieve its goal are those that are involved in a causal link with another action in the plan, and we collect those in w_{PT} as specified in Equation 1 (the plan cannot depend on any negative facts as they cannot appear in either the goal or in action preconditions). Basically, w_{PT} is the conjunction of those facts strictly required for executing the plan P (and, consequently, the devised template PT), and is used for devising a new problem $PR_{wp} = \{w_{PT}, goal_{PR}\}$. We can then drop the closed world assumption. For any initial state that satisfies w_{PT} , the obtained process template PT will be executable and achieve the goal condition $goal_{PR}$. **Example.** If we invoke `calcWP` on the causal links devised from P_1 , we may infer w_{PT_1} . Hence, for executing PT_1 (cf. Fig. 5(a)) we need to know the positions and capabilities of $act1$, $act2$, $act4$ and $rb1$; the other contextual information is not strictly needed for a correct execution of the template.

Thirdly, after the process template PT has been synthesized starting from P , it can be stored in our library together with information about the planning domain PD and abstracted problem PR_{wp} . Specifically, for every different planning domain PD devised through our approach, there is a pointer to a list of different abstracted planning problems PR_{wp} used for obtaining consistent plans in previous executions of our tool, together with the devised process templates. When a process designer defines a new domain theory D_{new} and a case C_{new} , the system checks if the corresponding planning domain PD_{new} and problem PR_{new} (obtained by applying f_{PC2PR} to D_{new} and C_{new}) are already present in our library. If the library contains a planning domain PD and an abstracted planning problem PR_{wp} (together with the associated template PT_{lib}) such that $PD_{new} = PD$ and $goal_{PR} = goal_{PR_{new}}$ and with $init_{PR_{new}} \vdash w_{PT}$, then PT_{lib} is executable respect to PR_{new} (and therefore with respect to C_{new}). This makes our templates reusable in a variety of different situations, in which we don't have complete information about the starting state. At this point, the process designer may decide to execute through an external PMS the template PT_{lib} just found, or to refine D_{new} and C_{new} if PT_{lib} does not fit with the designer expectations. **Example.** Let us suppose that the template shown in Fig. 5(a) does not satisfy at all the process designer, since s/he could add one further robot $rb2$ to the scenario in order to increase the degree of parallelism in the tasks execution. It follows that a new starting condition $init_{C_3}$ including also contextual information about $rb2$ can be defined. The associated initial planning state $init_{PR_3}$, together with the original goal condition $goal_{PR_1}$ and the planning domain PD_1 are first used for verifying if a previously executed template is already stored in the library. The library returns the template PT_1 shown in Fig. 5(a), since its weakest preconditions w_{PT_1} are satisfied by $init_{PR_3}$ (i.e., $init_{PR_3} \vdash w_{PT_1}$), and goal condition and planning domain are the same as before. Even if the template in Fig. 5(a) is executable with $init_{PR_3}$, the designer may try to search for another plan that (maybe) could exploit the presence of the new robot $rb2$. The planner builds a new plan starting from $init_{PR_3}$, and the associated template PT_2 is shown in Fig. 5(b). PT_2 requires the presence of one more robot (i.e., robot $rb2$) and more contextual information for being executed (so its weakest preconditions w_{PT_2} are "richer" than w_{PT_1}), but it provides an higher degree of

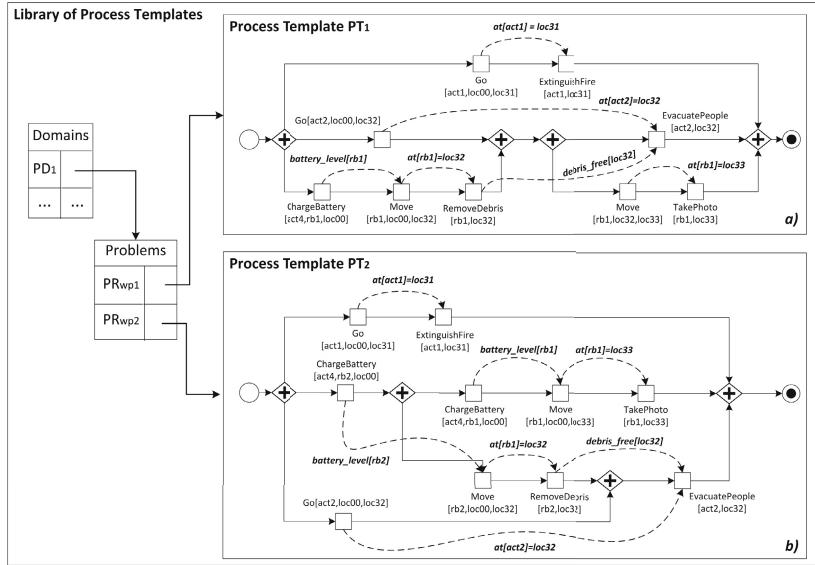


Fig. 5. Templates dealing with the scenario in Fig. 1

concurrency in the execution of its tasks. This means that the process designer can choose which template is the best for her/his purposes: one with less concurrency in the tasks enactment but with the fewest participants (cf., Fig. 5(a)), or one with more concurrency but requiring more resources for being executed (cf., Fig. 5(b)).

Despite the fact that a template is executable “as is”, it can be seen as an “intermediate version” of a completely defined process. In fact, the present POP-based tool cannot be used to synthesize templates involving loops or branching on conditions, and the designer may develop these manually by customizing the template to the specifics of the situation.

Properties. A process template PT guarantees some interesting properties, such as the *executability* of the template with respect to the information available in the starting state, and the property of *sound concurrency*, meaning that concurrent activities of a template are proven to be independent from each other.

Theorem 1. Given a solution plan P , a process template PT synthesized for P using our approach is **executable** for any process case C that satisfies the weakest preconditions wp_{PT} inferred from P .

The proof is straightforward. By definition, a sound planner generates a *consistent* plan [3] that leads from an initial state to a goal. Since we represent the Domain Theory/Case as PDDL planning domain/problem, the planner synthesizes a plan (i.e., a process template) that is executable with respect to Definition 7.

A second property we can prove is *sound concurrency*. Even if in a process designed by following data and workflow patterns [8] the concurrent execution of

two or more tasks should guarantee the consistency of data accessed by the concurrent tasks, in practice this is often not true. In fact, in complex environments there isn't a clear correlation between a change in the context and corresponding process changes, making it difficult to design by hand a process where concurrent tasks are also independent. On the contrary, all concurrent tasks of a template built with our approach are proven to be *independent* one from another.

Theorem 2. *Given a process template PT synthesized with our approach, all concurrent tasks are *independent*.*

Proof. By contradiction, let us suppose that a process template PT has two concurrent tasks t_1 and t_2 such that $t_1 \nparallel t_2$. Hence, t_1 (or t_2) has some effect affecting the precondition of t_2 (or of t_1). This means that $t_1 \triangleright t_2$ or $t_2 \triangleright t_1$. Since PT has been synthesized as result of a POP planner, this dependency between t_1 and t_2 would be represented with a causal link $t_1 \xrightarrow{e} t_2$ (or $t_2 \xrightarrow{e} t_1$), where e is an effect of task t_1 and a precondition for task t_2 (or vice-versa). This causal link requires an ordering between t_1 and t_2 , meaning they need to be executed (and represented in the process template) in sequence. But this means that t_1 and t_2 are not concurrent tasks, by contradicting the original hypothesis. \square

Experiments. To show the feasibility of our approach, we ran some experiments and measured the time required for synthesizing a partially ordered plan for some variants of our running example described in Section 2. We ran our tests using POPF2 [9] on an Intel U7300 1.30GHz, 4GB RAM machine. POPF2 is a temporal planner that handles PDDL 2.1 [4] and preserves the benefits of partial-order plan construction in terms of producing makespan-efficient, flexible plans.

The experimental setup was run on variants of our running example. We represented 7 planning actions in PD (corresponding to 7 different tasks stored in the tasks repository T), annotated with 7 relational predicates and 6 numeric fluents, in order to make the planner search space sufficiently challenging. Then, we defined 18 different planning problems of varying complexity by manipulating the number of facts in the goal. As well, we examined how irrelevant domain knowledge affects the performance of the planner. Starting from a planning problem PR with an initial state init_{PR} completely specified and with a goal condition goal_{PR} expressed as the conjunction of n facts, we manipulated the specification of the initial state init_{PR} to reduce the number of known facts. In our experiments, the number of facts in goal condition ranges from 1 single fact to a conjunction of 6 logical facts (that make the contextual problem harder). As shown in Table 1, for a given goal condition composed of n facts, our purpose was to measure the computation time needed for finding a sub-optimal solution for problems specified with starting states with a decreasing amount of knowledge. The column labeled as “Knowledge in init_{PR} ” makes explicit which information is removed from the initial state of the planning problem. For example, if we consider our running scenario from Section 2, whose goal condition is composed of 3 facts and characterized by a complete specification of the starting state, the time needed for finding a solution plan is of 0.13 seconds. After removing from the initial state all the information concerning the actor $act3$, the time

Table 1. Time performances of POPF2

Facts in $goal_{PR}$	Knowledge in $init_{PR}$	Time for a sub-opt. sol.
1	complete state	0.17
	No information about act1	0.15
	No information about act1 and act3	0.12
2	complete state	0.12
	No information about act3	0.10
	No information about act3 and rb1	0.08
3	complete state	0.13
	No information about act3	0.11
	No information about act3 and rb2	0.09
4	complete state	0.21
	No information about act3	0.20
	No information about act3 and rb1	0.10
5	complete state	0.17
	No information about act3	0.16
	No information about act3 and rb1	0.10
6	complete state	1.56
	No information about act3	1.19
	No information about act3 and act1	1.13

required for computing the plan decreases to 0.11 seconds. In general, for a given goal condition, removing “irrelevant information” from the initial state reduces the search space and the computation required for synthesizing the plan. Note that a sub-optimal solution includes more actions than those strictly required for fulfilling the goal, and when the number of facts in a goal condition increases, the quality of the solution may decrease. Based on our experiments, the approach seems feasible for medium-sized dynamic processes as used in practice.

6 Related Work

Process modeling is the first and most important step in the BPM lifecycle [1], which intends to provide a high-level specification of a business process that is independent from implementation and serves as a basis for process automation and verification. The task of defining a model is often performed with the aid of tools that provide a graphical representation, but without any automatic generation of the process model. However, in recent years, numerous AI planning-based approaches have been devised for the latter, and the closest to our approach are [10,11,12]. [10] presents the basic idea behind the use of planning techniques for generating a process schema, but no implementation seems to be provided, and the direct use of the PDDL language for specifying the domain theory requires a deep understanding of AI planning technology. In [11], the authors exploit the IPSS planner for modeling processes in SHAMASH [13], a knowledge-based system that uses a rule-based approach. To automate the process model generation, they first translate the semantic representation of SHAMASH into the IPSS language. Then, IPSS produces a parallel plan of activities that is finally translated back into SHAMASH and is presented graphically to the user. However, the emphasis is on supporting processes for which one has complete knowledge, while for dynamic processes some contextual information may not be available at the time of process model synthesis. The work of [12] is based

on learning activities as planning operators and feeding them to a planner that generates the process model. An interesting result concerns the possibility of producing process models even though the activities may not be accurately described. In such cases, the authors use a best-effort planner that is always able to create a plan, even though the plan may be incorrect. After a finite number of refinements, the best candidate plan (i.e., the one with the lowest number of unsatisfied preconditions) is translated into a process model. Unfortunately, the best plan found is often far from the correct solution [12].

7 Conclusion

In this paper, we developed a technique based on POP algorithms and declarative specifications of process tasks for synthesizing a library of process templates to be enacted in partially specified contextual scenarios. We are currently working on a complete implementation and thorough validation of the approach, including the formalization of metrics for evaluating process templates' quality. A future direction for this work is to generate hierarchical process templates, with high-level templates achieving more general goals that can invoke simpler templates to achieve some of their subgoals. We also plan to address expressiveness limitations, such as handling preferences and representing negative preconditions.

References

1. Weske, M.: Business Process Management: Concepts, Languages, Architectures. 2nd edn. Springer (2010)
2. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems. Springer, Berlin (2012)
3. Weld, D.: An Introduction to Least Commitment Planning. *AI Mag.* 15(4) (1994)
4. Fox, M., Long, D.: PDDL2.1: an Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Int. Res.* 20(1) (2003)
5. White, S.A., Miers, D.: BPMN Modeling and Reference Guide: Understanding and Using BPMN. Future Strategies Inc. (2008)
6. Godefroid, P.: Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem. Springer (1996)
7. Reiter, R.: On Closed World Data Bases. In: Ginsberg, M. (ed.) *Readings in Non-monotonic Reasoning*, Morgan Kaufmann Publishers Inc. (1987)
8. Dumas, M., van der Aalst, W.M.: Process-aware information systems: bridging people and software through process technology. Wiley-Interscience (2005)
9. Coles, A.J., Coles, A., Fox, M., Long, D.: Forward-Chaining Partial-Order Planning. In: ICAPS (2010)
10. Schuschel, H., Weske, M.: Triggering replanning in an integrated workflow planning and enactment system. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) *ADBIS 2004*. LNCS, vol. 3255, pp. 322–335. Springer, Heidelberg (2004)
11. R-Moreno, M.D., Borrajo, D., Cesta, A., Oddi, A.: Integrating Planning and Scheduling in Workflow Domains. *Exp. Syst. with App.*: An Int. J. 33(2) (2007)
12. Ferreira, H., Ferreira, D.: An Integrated Life Cycle for Workflow Management Based on Learning and Planning. *Int. J. Coop. Inf. Systems* 15 (2006)
13. Aler, R., Borrajo, D., Camacho, D.: A Knowledge-based Approach for Business Process Reengineering, SHAMASH. *Know.-Based Syst.* 15(8) (2002)

Spotting Terminology Deficiencies in Process Model Repositories

Fabian Pittke^{1,3}, Henrik Leopold¹, and Jan Mendling²

¹ Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany

henrik.leopold@wiwi.hu-berlin.de

² WU Vienna, Augasse 2-6, A-1090 Vienna, Austria

jan.mendling@wu.ac.at

³ SRH University Berlin, Ernst-Reuter-Platz 10, 10587 Berlin, Germany

fabian.pittke@srh-uni-berlin.de

Abstract. Thinking in business processes and using process models for their documentation has become common practice in companies. In many cases this documentation encompasses more than thousands of models. One of the key challenges is achieving consistency of the process model terminology. Especially, the usage of synonym and homonym words is one of the most severe problems for terminological consistency. Therefore, this paper presents an automatic approach to identify synonym and homonym words in model repositories. We challenged the approach against three model collection from practice that are assumed to have different levels of terminological consistency. The evaluation shows that the approach is capable to fulfill these goals and to identify meaningful synonym and homonym candidates for follow-up resolution.

Keywords: Identification of Synonyms, Identification of Homonyms, Business Process Models.

1 Introduction

Business process models have become an integral of general documentation available in enterprises, often covering thousands of models. A key challenge for such large-scale modeling initiatives is to achieve consistency and comparability [1] of the models, which are created by different process analysts or by the various business practitioners themselves. Without appropriate measures, models are often inconsistent in terms of their layout, their level of detail, their labeling styles, or their terminology [1,2,3].

The issue of inconsistent terminology has been acknowledged in various areas of conceptual modeling. It relates to the semantic level of a model and the meaning associated with elements and names of elements [4]. The usage of synonyms and homonyms is one of the most tangible symptoms of inconsistent terminology [5,6,7], for example when both words *invoice* and *bill* are used. Proposals like creating a domain thesaurus [8] or technical term modeling [9] are well justified to fix such terminology issues before starting to model. However, modelers might

find it too restrictive to check terms while modeling and tools might not be able to enforce term usage. Also, such restrictions do not directly help in cleaning up an existing model collection.

Against this background, we approach the problem of synonyms and homonyms from a non-restrictive perspective. We assume that modelers have complete control on how to assign names to model elements, which is indeed inline with how tools generally support model creation. In such a setting, automatic analysis capabilities are required to inspect a model repository for potential terminological problems. The contribution of this research is a technique for the automatic detection of synonyms and homonyms in a model repository. This technique is meant to be used either by repository managers in an offline mode or by process analysts in an online mode for spotting issues and providing recommendations for solving them. The capabilities of this technique are evaluated for three process model collections from practice.

This paper proceeds as follows. Section 2 illustrates the terminology problem and provides an overview of its theoretical background. Section 3 defines the concepts of our automatic detection technique. Section 4 presents the results of applying our technique for three process model collections from practice. Section 5 relates our contribution to other research in the area of conceptual modeling. Finally, Section 6 concludes the paper and gives an outlook on future research.

2 Problem Illustration

Different aspects of process model quality have been addressed in prior research. For instance, structural and behavioral problems can be automatically identified and resolved using verification techniques [10,11,12]. The natural language content of process models has only been considered to a limited extent. For instance, available refactoring techniques rework the grammatical structure of an activity label and extract the action and the business object [3]. However, process models in practice often exhibit terminological weaknesses. The particular challenge relates to the fact that one syntactical word can have multiple meanings (homonymy) and that the same meaning can be represented by the different syntactical words (synonymy). In linguistic literature this phenomenon of meaning is discussed in the field of *lexical semantics* [13].

The impact of word meaning ambiguity for process model quality is illustrated in Figure 1, showing two process models created by different modelers. Scenario A describes a job application process of a company, i.e. a person that is interested in a certain job position applies for it and sends his or her application to the company. Scenario B depicts a software development process. It starts with a pre-analysis of application requirements, before relevant requirements are identified and evaluated. Thereafter, the application is designed, implemented, tested and installed.

As highlighted in Figure 1, we observe that the word *application* is used in two different contexts leading to different meanings. In scenario A, the word *application* clearly refers to a written request for employment, while scenario B uses the word in the context of a software program. From a linguistic point of

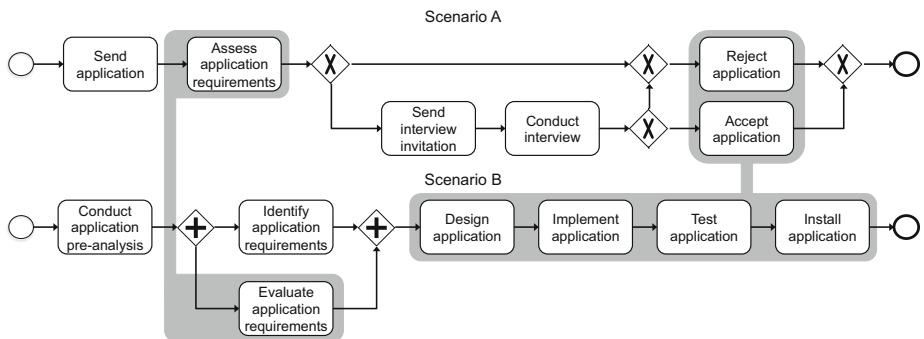


Fig. 1. Example of Business Process Models with Synonyms and Homonyms

view, the word *application* is a homonym, from which the according meaning has to be derived by the process modeler. The labels *Assess application requirements* from scenario A and *Evaluate application requirements* from scenario B further highlight the problem of homonyms. Both activities instruct employees to work on a business object represented by the syntactically identical word *application requirements*. Accordingly, process stakeholders cannot necessarily distinguish whether the requirements for a software application or requirements for a job position have to be evaluated. Again, the homonymous usage of the word application complicates the understanding of the process models and decreases the terminological clarity of these labels.

As depicted in Figure 1, the labels *Assess application requirements* and *Evaluate application requirements* both give the instruction to perform an evaluation task on the business object at hand. Furthermore, we notice that the actions *to assess* and *to evaluate* are used in the sense of *estimating the quality or significance* of the application requirements. Since the meaning of both actions is equal, both actions can be considered as synonym words. In consequence, the usage of synonym words also impedes the understandability of process models. As a solution, one could think of replacing one word with such a word that is more comprehensible to process stakeholders.

To identify such problems in business process models, we face the challenge of automatically determining the meaning of words. As illustrated in the example above, that can be very challenging. The word *application requirements* is used in two completely different contexts and there is only sparse information available to correctly recognize this fact. Since process models only contain short linguistic fragments which in many cases do not even represent proper sentences, it is not possible to directly use standard disambiguation technology from linguistics [14,15,16]. Techniques for word meaning disambiguation usually employ statistical methods working on the syntactic structure of the underlying sentences. Hence, if this syntactic structure of sentences is missing or cannot be recognized, these techniques cannot be applied. In order to still identify synonyms and homonyms in process models, alternative strategies have to be pursued. However, given these examples,

it is apparent that the identification of synonyms and homonyms is an important step for improving the overall quality of a model collection.

3 Conceptual Approach

This section introduces the conceptual approach for the identification of synonyms and homonyms in process models. We first summarize preliminaries before illustrating the steps to identify synonyms or homonyms respectively.

3.1 Preliminaries

We start with the definition of the objects subject to this research, i.e. the activity labels of process models in general. According to [17], process model activities may follow different label styles, meaning that the same information can be conveyed in different ways. As an example consider the label *Notify customer* containing the action *to notify* as a verb and the business object *customer* as a noun. The same information could be also provided with the label *Customer notification*, where the action *to notify* is represented by the noun *notification*. However, since the technique defined in [3] is capable of aligning different labeling styles and automatically identifying action and business object of a label, we abstract from these problems in the remainder of the paper. As a result, we can directly work with the actions and business objects of the considered activities.

We start with the set of activities \mathcal{A} derived from a model collection. For each activity $a \in \mathcal{A}$ we consider one action a_{action} and one business object a_{bo} . Due to simplicity, we abstract from multiple actions and business objects here. Further, we define $words_{action}$ as the set of actions of a process model collection, i.e. $words_{action} = \bigcup_{a \in \mathcal{A}} a_{action}$. The set $words_{bo}$ comprises all business objects, i.e. $words_{bo} = \bigcup_{a \in \mathcal{A}} a_{bo}$. Finally, the union of both sets $\mathcal{W} = words_{action} \cup words_{bo}$ defines the set of all words of a given collection. In the example of Figure 1, the words *application*, *application requirements* or *interview invitation* belong to $words_{bo}$, while the words *to send*, *to assess* or *to identify* are accordingly part of $words_{action}$.

Additionally, let \mathcal{M} be the set of all meanings and $m_i \in \mathcal{M} (i = 1, \dots, n)$ all meanings that one syntactical word $w \in \mathcal{W}$ is associated with. In order to capture the relation between meanings and a syntactical word, we define the relation *Word Meaning* \mathcal{WM} that matches one word with its corresponding meanings:

$$\mathcal{WM} \subseteq \mathcal{W} \times \mathcal{M} \quad (1)$$

As an example, consider the word *to send*. According to the lexical database WordNet [18], that word has eight different meanings. One of this meanings is described as *cause to be directed or transmitted to another place*. For this particular meaning, synonyms such as *to mail* or *to post* can be retrieved. However, if *to send* is used in the sense of *sending something over the airwaves*, the words *to broadcast* or *to air* would be more suitable synonyms. This example makes clear that lexical relations

such as synonymy must be identified based on the word meaning and cannot be simply defined by looking at a syntactical sequence of characters.

Accordingly, the synonym and the homonym relation is defined. A *synonym relation* is a symmetric relation between two words $w_1, w_2 \in \mathcal{W}$, where both words represent the same meaning $m \in \mathcal{M}$. Formally, the synonym relation can be defined as follows:

$$Syn = \{(w_1, w_2) | \exists_{m \in \mathcal{M}} : (w_1, m) \in \mathcal{WM} \wedge (w_2, m) \in \mathcal{WM}\} \quad (2)$$

The *homonym relation* is a relation of one word to its meanings, where this word represents two different meanings $m_1, m_2 \in \mathcal{M}$. The relation can formally be described as follows:

$$Hom = \{w \in \mathcal{W} | \exists_{m_1, m_2 \in \mathcal{M}} : (w, m_1) \in \mathcal{WM} \wedge (w, m_2) \in \mathcal{WM}\} \quad (3)$$

Considering the process models in Figure 1 we note that both process models use the business object *application*. Yet, we observe that the business object differs in its meanings. In scenario A, the word *application* is used in the context of a written request for employment. In scenario B, *application* refers to a software application. Hence, the context could actually reveal that we face two different meanings m_1 and m_2 for a syntactically identical word. Therefore, definition 3 classifies the word *application* as a homonym.

Aiming for the automated identification of such phenomena, it is crucial to appropriately determine the meanings which can be associated with a given word. For identifying all possible meanings of a word, we can use lexical databases such as WordNet [18]. WordNet organizes words in sets of synonyms, so called Synsets. Each Synset represents a certain meaning of a given word. One word can occur in multiple Synsets. Among others, Wordnet also defines direct lexical relations between words, such as synonymy or meronymy (*part of* relationship). Homonymy is not directly associated with a word, but can be assessed based on the number of Synsets associated with it.

In general, the meaning of a word depends upon the context of its use. In natural language texts, context is given by one or more sentences or a paragraph respectively. Therefore, linguistic approaches often make use of a large context from which the meaning of a word can be derived. In process models, however, context information is very sparse. The symbols and the small pieces of texts are not sufficient for standard natural language techniques to derive the meaning of a given word. In order to still be able to select a likely meaning, we make use of the words co-occurring in the label. For an activity label such as *Process application*, this is the action *to process* and the business object *application*. Although this is limited information, the action *process* can already help to reduce the potential number of meanings which can be associated with *application*.

Accordingly, we introduce the notion of context in process models. First, we define the context for an activity $a \in \mathcal{A}$ containing the word $w \in \mathcal{W}$. As previously defined, the word w can either represent an action or a business object.

The function *labelContext* returns the corresponding action, if w represents a business object and returns the business object if w is an action:

$$\text{labelContext}(w, a) = \begin{cases} a_{\text{action}} & \text{if } w \in \mathcal{W}_{\text{bo}} \\ a_{\text{bo}} & \text{if } w \in \mathcal{W}_{\text{action}} \end{cases} \quad (4)$$

Second, we define the context of a word w in a process model that consists of a specific set of activities \mathcal{A} . The function *modelContext* returns the set of words which contextualize the input word w for a set of activities \mathcal{A} typically stemming from the whole repository:

$$\text{modelContext}(w, \mathcal{A}) = \{\text{labelContext}(w, a) | a \in \mathcal{A}\} \quad (5)$$

3.2 Identification of Synonyms

Considering the definition from Equation 2, the identification of synonym words is reduced to the identification of word pairs with a common meaning. Therefore, semantic metrics based on WordNet are employed to identify synonyms [19]. Among them, the Lin measure correlates best with human judgment [20]. Yet, these metrics are of limited use, because there is no differentiation between synonym and similar words. For example, consider the words (*create, produce*) that are synonyms in the sense of manufacturing a product. The Lin value only amounts to 0.64 and does not suggest a synonym relation. The pair (*make, produce*) even scores 0.0 and would not be considered at all. Another issue is that the procedure must be operationalized for process models. The pairwise comparison of words is not efficient for large process model collections as the consideration of all possible word combinations results in a huge search space. Further, it is necessary to adequately determine the meaning of the considered words. Otherwise, if the context of a word is not considered, the result set would suffer from a low precision.

To clarify how we address these challenges in the proposed approach, consider the example from Figure 1. To reduce the overall search space and guarantee that a considered word pair is semantically related, the approach only investigates word pairs with a common *context word*. That means that two actions are only considered as potential synonym candidates, if they are both applied to the same business object. In Figure 1 this holds for the actions *to assess* and the *to evaluate* of the labels *Assess application requirements* and *Evaluate application requirements* as they both share the common business object *application requirements*. In order to verify the assumption of synonymy, WordNet can be used to check, if these actions share a common meaning. In case of a common context and a common meaning in WordNet, two words are consequently considered as synonyms.

Algorithm 1 illustrates the required steps formally. The algorithm starts with the initialization of the result set. Afterwards, the algorithm determines the set of words *modelWords* from the collection given a context word w (lines 3–4). For each pair $w_1, w_2 \in \text{modelWords}$, the meaning is identified and stored in separate sets (lines 5–8). If at least one meaning is identified that is shared by both words, these words are stored in the result set (lines 9–10). The algorithm terminates with the return of the set of potential synonym candidates (line 11).

Algorithm 1. Identification of Synonyms in a Process Repository

```

1: identifySynonyms(Activities  $\mathcal{A}$ )
2: Set  $synonymCandidates = \text{new List}()$ ;
3: for all  $w \in \mathcal{W}$  do
4:   Set  $modelWords = \text{getModelContext}(w, \mathcal{A})$ ;
5:   for all  $w_1 \in modelWords$  do
6:     for all  $w_2 \in modelWords$  do
7:       Set  $s_1 = \text{wordnet.getSynsets}(w_1)$ ;
8:       Set  $s_2 = \text{wordnet.getSynsets}(w_2)$ ;
9:       if  $s_1 \cap s_2 \neq \emptyset$  then
10:          $synonymCandidates.add(w_1, w_2)$ ;
11: return  $synonymCandidates$ ;

```

3.3 Identification of Homonyms

As outlined before, the identification of homonyms relates to the problem to find words in a process model collection having different meanings in diverging contexts. Following the formal definition of a homonym (see Equation 3), words have to be identified which have at least two different meanings. However, as many words have slightly varying meanings, this would be neither efficient nor effective. The challenge at hand is to identify words which do not only have different meanings, but are also used with different meanings.

In order to address this problem, we apply the function $modelContext$ on a given word to obtain the context from all process models containing the target word. Using the SenseRelate approach [21] the context can be used to disambiguate the target word by finding the most fitting meanings. Yet, this does not properly identify homonyms, since one word can have different meanings that are semantically very close.

As example consider the word *insurance*. According to WordNet an insurance can be a promise of reimbursement in the case of loss, a written contract or certificate of insurance, or the protection against future loss. Obviously, these meanings are closely related with the result that *insurance* cannot be considered as potential homonym. Therefore, a refinement is needed that supports the distinction between more or less ambiguous homonyms. This is done by the *Semantic Homogeneity* indicator SH . It takes several information as an input, i.e. the target word w , the Synsets of the target word and a weight for each Synset. This weight is calculated from SenseRelate that determines the average similarity scores of the target word to the respective Synset given a context. We denote the weight of a Synset as w_i . The SH indicator is then calculated as the weighted sum of similarity scores from the target word to each word of the respective Synset, denoted as $sim(w, S_i)$ (see Equation 6). The indicator ranges from 0 to 1. A score closer to 0 indicates that the word is used with different meanings and therefore is likely considered as homonym, while higher scores up to 1 classify a word as non-homonym.

Algorithm 2. Identification of Homonyms in a Process Repository

```

1: identifyHomonyms(Activities  $\mathcal{A}$ )
2: Map  $homonyms = \text{new Map}();$ 
3: for all  $w \in \mathcal{W}$  do
4:   List  $wordContext = \text{getModelContext}(w, \mathcal{A});$ 
5:   Map  $weightedSynsets = \text{SenseRelate.getWeightsForSynsets}(w, wordContext);$ 

6:   if  $weightedSynsets.size() > 1$  then
7:     float  $SH = \text{calculateSemanticHomogeneity}(w, weightedSynsets);$ 
8:      $homonyms.add(w, SH, weightedSynsets);$ 
9: return  $homonyms;$ 

```

$$SH(w, \mathcal{S}) = \sum_{i=1}^n w_i \frac{\text{sim}(w, S_i)}{|S_i|} (\forall S_i \in \mathcal{S}) \quad (6)$$

Algorithm 2 formalizes the homonym identification approach. The algorithm starts with the initialization of the result map that stores the semantic homogeneity score as well as the set of most probable Synsets for each word (line 2). For each word, we extract all context words and calculate the weight for each Synset using SenseRelate. The result is stored in a map (lines 3–5). If it turns out that a word only has one Synset, we can exclude this word to be a homonym according to the definition. Otherwise, we calculate the semantic homogeneity (line 6–7). Finally, the semantic homogeneity is stored in the result map along with the word and the weighted Synsets (lines 8). The algorithm terminates by returning the set of potential homonym candidates (line 9).

3.4 Resolving Terminological Issues

The presented identification techniques provide sets of words that are used as synonyms or homonyms in the analyzed model collection. Both analysis results are considered as terminological issues that have to be resolved in the next step. Accordingly, repository managers can use the identified cases to manually resolve these issues. For synonym words, the repository manager can replace a considered word with the a specific word. In case of our example, the repository manager might replace the action *to assess* with the action *to evaluate*, since the latter one is more specific according to WordNet. This replacement can potentially be conducted automatically. For homonym resolution, the repository manager might add additional words to disambiguate the identified homonyms. For our example, one might change the business object *application* to *job application* in scenario A and to *software application* in scenario B. As a result of applying these resolution techniques, the specificity and the terminological quality of the repository will be increased and the understandability can be improved.

4 Evaluation

The presented identification techniques are challenged against process model repositories from practice. The evaluation takes three different repositories into account that have different characteristics with respect to terminological quality. Section 4.1 provides detailed information on each repository, whereas Section 4.2 presents the results of the identification techniques. We further provide examples of identified synonyms or homonyms respectively.

4.1 Model Repository Demographics

In order to demonstrate the applicability of the presented techniques, we employ three different model collections from practice. We selected collections differing in the expected degree of terminological standardization. Since there is no gold standard available, we test whether our techniques are capable of identifying the assumed degree of homonymy and synonymy in the collections. Accordingly, we expect to find more synonyms and homonyms in non professionally maintained repositories containing models from a large number of modelers. Table 1 provides an overview of characteristics of each repository.

The SAP Reference Model contains 604 Event-Driven Process Chains organized in 29 different functional branches [22]. Examples are procurement, sales or financial accounting. The model collection includes 2433 activity labels with 322 unique actions and 885 unique business objects. Since the SAP Reference Model was designed as a recommendation for the industry using a standard terminology, we expect a small number of homonyms and synonyms.

The process model collection from an international telecommunication company, which we will refer to as TelCo, comprises 286 separate process models with 3155 activities. We identified 557 distinct actions and 1959 business objects to apply the approach. We assume TelCo to be less strictly standardized as it uses terminology for telecommunication industry. However, there is no central glossary of terms available as in the case of the SAP models.

The model collection from the BPM Academic Initiative (AI)¹ comprises tens of thousands of process models. The models are formalized in various modeling languages and size. Our subset includes 597 process models, mostly in BPMN notation from a wide range of industrial and academic institutions. The collection subset encompasses 4958 activity labels in total, where 1200 actions and 3132 business objects are distinct. Since the collection targets no specific industry and is rather uncontrolled, the number of synonyms and homonyms is expected to be the highest among all repositories.

4.2 Evaluation Results

This section presents the results of all model repositories. According to the assumptions for each model repository, we check whether the identification technique is capable to identify the overall tendency of the respective collection.

¹ <http://bpmai.org>

Table 1. Details of Evaluation Sample

Characteristic	SAP	TelCo	AI
Models	604	286	597
Labels	2433	3155	4958
Unique actions	322	557	1200
Unique Business objects	885	1959	3132

Table 2. Results of Synonym Identification

Characteristic	SAP	TelCo	AI
Actions covered in WordNet	263 (81,7%)	283 (50,1%)	526 (43,8%)
Business Objects covered in WordNet	160 (18,1%)	226 (11,54%)	448 (14,3%)
Synonym actions	2	32	53
Synonym Business objects	2	12	102
Synonyms total	4	44	155

Thus, we show the absolute numbers of identified synonyms and homonyms for collections, before providing a list of commonly identified synonyms or homonyms for each collection.

Table 2 depicts the number of identified synonyms in the model collections. Since we rely on WordNet, we can only identify words that are part of the database. For SAP, 263 actions and 160 business objects are covered. Yet, most of the business objects, such as *transfer time sheet* or *customer scheduling agreement* are not included due to their specificity. The synonym identification approach has found 4 cases of synonym actions or business objects. We conclude that the SAP collection has a defined terminology for business processes that is used consistently and clearly supports the assumption we stated in the last section.

For TelCo, WordNet covers 283 action and 226 business objects. The approach identified 32 cases of interchangeable actions and 12 cases of business objects. Compared to SAP, the overall number of synonyms is higher. Nevertheless, we can assume that this industry uses a defined set of terms that is also reflected in their process models and that correspond to the assumption from Section 4.1.

The AI collection contained 526 actions and 448 business objects that could be analyzed resulting in 53 synonym actions and 102 business objects. Compared to SAP, we observe that these business processes are not modeled distinctively. Since many models stem from academic training and from different application areas, the degree of ambiguity is considerably higher. The high number of synonym word pairs confirms the initial assumption.

Figure 2 illustrates the quantitative results for homonym detection. Using the indicator of *Semantic Homogeneity* the numbers of homonym candidates for different ranges are depicted.

For SAP, the technique analyses 381 of 423 words. The loss of 42 words results from labels consisting of an action without any business object, i.e. having no context. This is similar for the other two collections (TelCo: 477 of 509; AI: 924

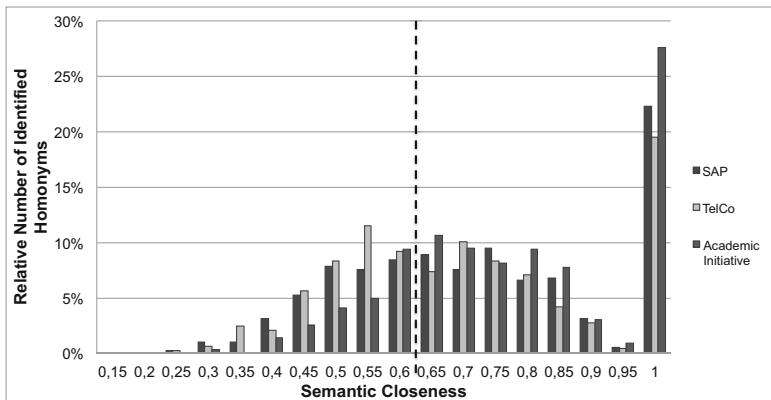


Fig. 2. Results of Homonym Identification

of 974). As depicted in Figure 2, the number of homonyms is steadily increasing until 0.6. Then, the score remains constant and decreases for values higher than 0.8. Besides, about 23% of words having a semantic homogeneity of 1.0 that denotes unambiguity. We observe a similar trend for TelCo. Again, nearly 20% of words that are unambiguous. For the AI collection, we observe an increase until 0.65 and a immediate decrease until 0.95. For an SH score of 1 the relative number is significantly higher for all three collections. By far, AI outperforms the other collections (27%), followed by SAP (23%) and TelCo (19%).

Having a closer look at the relative numbers of SAP and TelCo, we observe that TelCo exceeds SAP in the range from 0.2 to 0.6. For higher ranges, we observe the opposite. As a result, we conclude that SAP has a higher terminological quality for process models than TelCo which clearly corresponds to the assumption in Section 4.1. Yet, it appears that AI is far more precise compared to SAP. The average number of ambiguous words is smaller, whereas the average number of unambiguous words is higher, especially for a score of 1. Thus, the assumption stating that the AI is more ambiguous than SAP has to be rejected for homonyms. An explanation for this phenomenon might be that most of the unambiguous words, such as *transfer time sheet* or *customer scheduling agreement*, are not part of WordNet. Yet, we observe that for instance the word *transfer time sheet* is a meronym of the word *time sheet* that already is unambiguous. In consequence, we could conclude that the whole word is also unambiguous.

To conclude the evaluation, qualitative results of the identification techniques are provided. First, Table 3 depicts an extract of prominent examples of synonyms spread among all model collections. Frequently, synonym pairs in conjunction with the action *to send* are identified. Interchangeable words are *to ship* or *to transport*, if physical goods have to be delivered somewhere. In case messages have to be send, one might prefer actions such as *to transmit* or *to post*. The most prominent examples of synonym business objects are the pairs *client* and *customer* as well as *bill* and *invoice*.

Table 3. List of most frequent Synonyms

Rank	Synonym Actions	Frequency	Synonym Business Objects	Frequency
1	(place, send)	35	(client, customer)	14
2	(send, ship)	20	(bill, invoice)	12
3	(issue, release)	8	(data, information)	4
4	(send, transmit)	6	(inventory, stock)	3
5	(post, send)	6	(case, event)	2

Second, the Tables 4 and 5 show the results for homonym actions and business objects along with the respective SH score and their frequency in the evaluation sample.

By far, the action *to expire* has the lowest SH score. Although it appears only four times in the sample and although it only has three different meanings, the meaning *to pass from life* is dominating in the context. The proper meaning in business context is on the last position. In contrast, the action *to specify* has 7 different meanings and a similar SH score. Yet, the meaning in a business context only comes on the fourth position. We also listed the action *to time* as example with SH score equal to 1. Although this action has 5 different meanings, we observe that the different meanings are homogeneous with respect to assign a fixed time for an event in the future.

For the business objects, the approach identifies the word *go* as highly ambiguous ($SH = 0.38$) which is also reflected by the four different meanings of this word. Interestingly, the meaning with the highest weight is from the area of board games followed by a designer name for MDMA. It also seems that none of the meanings fits into a business context. Another interesting example is the business object *issue*. The word has 11 different meanings and appears that meaning 2 and 3 are prominent in business process context. It underlines the ambiguity of the word and the need for an unambiguous word such as supply or topic/subject as an alternative. A fine example for an unambiguous word is the business object *communication* that has three different yet similar meanings.

5 Related Work

This research relates to three major streams of research: approaches for automatically assuring process model quality, applications of natural language processing techniques in process models, and the field of lexical disambiguation.

Automatic quality assurance for process models has been intensively studied for structural properties. For example, soundness as a structural property of the state space can be efficiently checked using Petri-Net concepts [23,24]. The degree of structuredness can be improved by using automatic refactoring techniques [25,26]. Also the content of activity labels can be automatically refactored based on label style parsing techniques which are based on natural language processing [3,27]. Our technique complements these approaches with an automatic technique for detecting terminological issues.

Table 4. List of Homonym Actions

Word	<i>SH</i>	Frequency	Possible Meaning	Weight
expire	0.31	4	pass from physical life	0.66
			expel air	0.19
			lose validity	0.14
specify	0.33	7	be specific about	0.26
			determine the essential quality	0.25
			select something for a specific purpose	0.13
			specify as a condition or in an agreement	0.12
...
time	1	2	measure the time of an event	0.28
			adjust so that a force is applied and an action occurs at the desired time	0.23
			regulate or set the time of	0.16
			assign a time for an activity or event	0.16

Table 5. List of Homonym Business Objects

Word	<i>SH</i>	Frequency	Possible Meaning	Weight
go	0,38	6	a board game for two players who place counters on a grid	0,36
			street names for methylenedioxymethamphetamine	0,25
			a usually brief attempt	0,21
			a time for working (after which you will be relieved by someone)	0,17
issue	0,39	6	a phenomenon that follows some previous phenomenon	0,18
			act of providing an item	0,12
			some situation or event that is thought about	0,11
			the immediate descendants of a person	0,10
		
communication	1	2	the activity of communicating	0,43
			something that is communicated by people	0,41
			a connection allowing access between persons	0,14

Our technique also relates to the application of natural language techniques in conceptual modeling more generally. Examples are automatic service identification [28,29], the identification of semantically equivalent activities in different models [30,31], and the discovery of process patterns [32]. Our technique might serve as a preprocessing before applying these approaches.

The problem of disambiguation is one of the key problems of computational linguistics, with the approach by Sanderson being one of the most prominent ones [14]. However, short language fragments remains a current challenge [15,16].

Although we do not provide a general solution for this linguistic challenge, we operationalized and successfully address the issue in the context of process models.

6 Conclusion

We presented a novel approach for the automatic identification of synonym and homonym terms in process model repositories. The approach employs the meaning of terms based on the process model context as well as language processing techniques to identify homonyms and synonyms. Our techniques have been implemented prototypically and evaluated with more than 1400 process models from three different repositories from practice. The evaluation demonstrates its capability to spot and quantify terminological issues in these repositories.

In future research, we first plan to improve the approach by extending the search space of words and overcoming the limitations of WordNet. The goal is to cover more actions and business objects and identify more terminological issues. Second, we aim to extend the technique to resolve the identified synonyms and homonyms. We identified the context of a term especially in process models as a critical component and thus want to integrate additional business knowledge incorporated in specific domain ontologies or text corpora. We consider the integration of such knowledge-intensive technologies as a promising step to support the identification as well as the resolution of synonyms and homonyms.

References

1. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of Business Process Modeling. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) *Business Process Management*. LNCS, vol. 1806, pp. 30–49. Springer, Heidelberg (2000)
2. Davis, R.: *ARIS design platform: advanced process modelling and administration*. Springer (2008)
3. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. *Information Systems* 37(5), 443–459 (2012)
4. Thalheim, B.: Syntax, semantics and pragmatics of conceptual modelling. In: Bouma, G., Ittoo, A., Métais, E., Wortmann, H. (eds.) *NLDB 2012*. LNCS, vol. 7337, pp. 1–10. Springer, Heidelberg (2012)
5. Dean, D., Lee, J., Orwig, R., Vogel, D.: Technological support for group process modeling. *Journal of Management Information Systems*, 43–63 (1994)
6. Rosemann, M., Muehlen, M.: Evaluation of workflow management systems—a meta model approach. *Australian Journal of Information Systems* 6, 103–116 (1998)
7. Rolland, C.: L'e-lyee: coupling l'ecritoire and lyeeall. *Information & Software Technology* 44(3), 185–194 (2002)
8. Becker, J., Delfmann, P., Herwig, S., Lis, L., Stein, A.: Formalizing linguistic conventions for conceptual models. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) *ER 2009*. LNCS, vol. 5829, pp. 70–83. Springer, Heidelberg (2009)
9. Becker, J., Kugeler, M., Rosemann, M.: *Process management: a guide for the design of business processes*. Springer (2003)

10. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 278–293. Springer, Heidelberg (2009)
11. van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.F.: Process mining and verification of properties: An approach based on temporal logic. In: Meersman, R., Tari, Z. (eds.) CoopIS/DOA/ODBASE 2005, Part I. LNCS, vol. 3760, pp. 130–147. Springer, Heidelberg (2005)
12. van der Aalst, W.M.P., de Medeiros, A.K.A.: Process mining and security: Detecting anomalous process executions and checking process conformance. *Electron. Notes Theor. Comput. Sci.* 121, 3–21 (2005)
13. Cruse, A.: Meaning in Language: An Introduction to Semantics and Pragmatics. Oxford University Press (2004)
14. Sanderson, M.: Word sense disambiguation and information retrieval. In: ACM SIGIR 1994, pp. 142–151 (1994)
15. Stokoe, C., et al.: Word sense disambiguation in information retrieval revisited. In: ACM SIGIR, pp. 159–166 (2003)
16. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *JAIR* 11, 95–130 (1999)
17. Mendling, J., Reijers, H.A., Recker, J.: Activity Labeling in Process Modeling: Empirical Insights and Recommendations. *Inf. Sys.* 35(4), 467–482 (2010)
18. Miller, G.: WordNet: a Lexical Database for English. *CACM* 38(11), 39–41 (1995)
19. Ferret, O.: Testing Semantic Similarity Measures for Extracting Synonyms from a Corpus. In: LREC 2010, pp. 3038–3343. ELRA (2010)
20. Lin, D.: An information-theoretic definition of similarity. In: ICML 1998, pp. 296–304. Morgan Kaufmann (1998)
21. Patwardhan, S., Banerjee, S., Pedersen, T.: Senserelate:targetword: a generalized framework for word sense disambiguation. In: ACL 2005, pp. 73–76 (2005)
22. Keller, G., Teufel, T.: SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping. Addison-Wesley (1998)
23. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on Demand: Instantaneous Soundness Checking of Industrial Business Process Models. *Data & Knowledge Engineering* 70(5), 448–466 (2011)
24. van der Aalst, W.M.P., Hirnschall, A., Verbeek, H.M.W.: An alternative way to analyze workflow graphs. In: Piddock, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 535–552. Springer, Heidelberg (2002)
25. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. *Computers in Industry* 62(5), 467–486 (2011)
26. Polyvyanyy, A., García-Bañuelos, L., Dumas, M.: Structuring Acyclic Process Models. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 276–293. Springer, Heidelberg (2010)
27. Becker, J., Delfmann, P., Herwig, S., Lis, L., Stein, A.: Towards Increased Comparability of Conceptual Models - Enforcing Naming Conventions through Domain Thesauri and Linguistic Grammars. In: ECIS 2009, pp. 2231–2242 (2009)
28. Leopold, H., Mendling, J.: Automatic derivation of service candidates from business process model repositories. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS 2012. LNBIP, vol. 117, pp. 84–95. Springer, Heidelberg (2012)
29. Knackstedt, R., Kuropka, D., Müller, O.: An ontology-based service discovery approach for the provisioning of product-service bundles. In: ECIS 2008 (2008)

30. Becker, J., Breuker, D., Delfmann, P., Dietrich, H.A., Steinhorst, M.: Identifying business process activity mappings by optimizing behavioral similarity. In: AMCIS 2012 (2012)
31. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 319–334. Springer, Heidelberg (2012)
32. Gacitua-Decar, V., Pahl, C.: Automatic business process pattern matching for enterprise services design. Services II, 111–118 (2009)

Modeling of Reference Schemes

Terry Halpin

INTI International University, Malaysia
t.halpin@live.com

Abstract. In natural language, one usually refers to objects by means of proper names or definite descriptions. Languages for data modeling and/or ontology modeling vary considerably in their support for such natural reference schemes. An understanding of these differences is important both for modeling reference schemes within such languages and for transforming models from one language to another. This paper provides a critical and comparative review of reference scheme modeling within the Unified Modeling Language (version 2.5), the Barker dialect of Entity Relationship modeling, Object-Role Modeling (version 2), relational database modeling, and the Web Ontology Language (version 2.0). We identify which kinds of reference schemes can be captured within specific languages as well as those reference schemes that cannot be. Our analysis covers simple reference schemes, compound reference schemes, disjunctive reference and context-dependent reference schemes.

1 Introduction

If an object (in the sense of an individual item) is currently in view, one may refer to it by ostension (pointing at the object). In normal communication however, one typically refers to an object by using a linguistic expression. This allows one to reference concrete objects in view (e.g. oneself) or not currently in sight (e.g. the philosopher Plato), as well as intangible objects (e.g. a university course). In natural language, proper names (e.g. “Australia”) and definite descriptions [2] (e.g. “the prime minister of Australia”) are the kinds of referential, linguistic expression that correspond most closely to the way objects are referenced within computerized information systems. Within a given universe of discourse (UoD), use of proper names or definite descriptions to identify an object entails that they cannot refer to other objects in that UoD.

This paper provides a comparative review of how such reference schemes are supported in current versions of the following languages for data modeling or ontological modeling: the Unified Modeling Language (UML) [23], the Barker dialect of Entity Relationship modeling (Barker ER) [4], Object-Role Modeling (ORM) [11], relational database modeling, and the Web Ontology Language (OWL) [29]. An understanding of the significant differences in the way these languages support reference schemes is important both for modeling identification schemes within such languages and for transforming models from one language to another.

For conceptual data modeling, fact-oriented approaches such as ORM, Natural Language Information Analysis Method (NIAM) [31] and Fully Communication

Oriented Information Modeling (FCO-IM) [3] differ from ER and class modeling in UML by uniformly modeling elementary facts as unary or longer relationships that are instances of *fact types* (e.g. Person smokes, Person was born on Date, Person played Sport for Country) instead of modeling some facts as relationships and others as instances of attributes (e.g. Person.isSmoker, Person.birthdate). This attribute-free approach enables all facts, constraints, and derivation rules to be verbalized naturally in sentences easily understood and validated by nontechnical business users using concrete examples, and promotes semantic stability, since one never needs to remodel existing structures in order to add facts about attributes [13]. Overviews of various fact-oriented modeling approaches may be found in [12, 15], and a detailed coverage of ORM in [18]. OWL is also attribute-free, but OWL facts are restricted to binary relationships (subject-predicate-object triples) whose subjects cannot be literals.

The rest of the paper is structured as follows. Section 2 covers simple reference schemes in which an object is identified either by an individual constant or by means of a single attribute or a relationship to another object. Section 3 examines compound reference schemes, where an entity is identified by means of a combination of two or more attributes or relationships. Section 4 discusses disjunctive reference schemes, in which some components of the reference scheme are optional, as well as context-dependent reference schemes, where the preferred identifier for an entity varies according to the context. Section 5 concludes by summarizing the main contributions, identifying areas for future research, and listing the references cited.

2 Simple Reference Schemes

In ORM, an *object* is any individual thing of interest, so corresponds to the notion of individual in classical logic. Figure 1(a) displays a simplified ORM metamodel fragment that partitions object types into entity types, domain value types and datatypes. ORM displays object types as named, soft rectangles using solid, dashed or dotted lines respectively for entity types, domain value types and datatypes. Subtyping relationships are depicted by solid arrows from subtype to supertype. The lifebuoy symbol denotes an exclusive-or constraint.

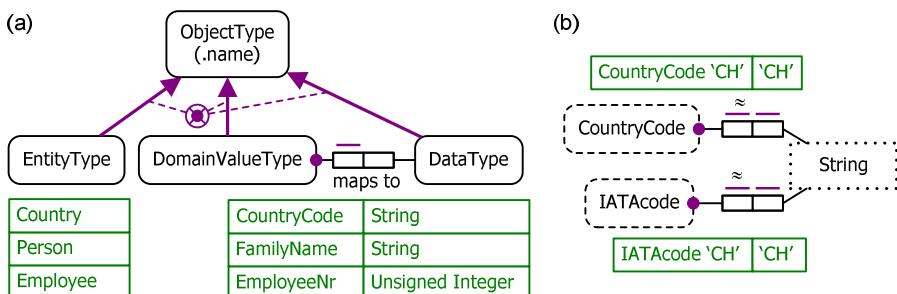


Fig. 1. ORM viewpoint on (a) object types and (b) value representation

An ORM *fact* either asserts the existence of an object, or predicates over one or more objects. A *fact role* (or role for short) is a part played within a fact relationship, and is depicted as a box connected by a line to its object type. An *elementary fact type* is a set of one or more typed, logical predicates for expressing one kind of irreducible fact. A *predicate* is shown as an ordered set of role boxes, with a predicate reading (read left to right or top-down unless reversed by an arrow tip). A bar beside a role depicts a uniqueness constraint (for each state of the model, each instance playing that role appears only once). A solid dot attached to an end of a role connector depicts a mandatory role constraint (each instance in the current population of the role's type must play that role). Sample instances of the object types and fact types in Figure 1 appear in the fact tables shown.

In ORM, an *entity* is an object that has an explicit reference scheme involving at least one relationship and that typically may change its state (e.g. the Country that has the CountryCode ‘CH’). A *domain value* (also called a semantic value, or simply a value) is essentially a typed constant (e.g. the CountryCode ‘CH’), and a *data item* (or data value) is an instance of a datatype (e.g. ‘CH’). For simplicity, finer aspects of data types such as facets (e.g. assigning country codes a fixed length of 2 characters) and language culture (e.g. en:US) are ignored in this paper.

ORM’s distinction between domain values and data values is needed to conform to the Principle of Indiscernibility of Identicals (identical objects have exactly the same properties). Using Φ as variable ranging over predicates, this may be formalized in second-order logic as follows: $\forall x,y [x = y \rightarrow \forall \Phi (\Phi x \equiv \Phi y)]$. For example, the country code ‘CH’ is based on a term in Latin (*Confederatio Helvetica*, the Latin name for Switzerland), but this is not true of the IATA airline code ‘CH’ (used for Bemidji airlines), so the country code ‘CH’ and the IATA code ‘CH’ are not identical, even though they are represented by the same character string ‘CH’. Our latest formalization of ORM [16] reserves the predicate symbol \approx for the representation relationship that provides an injective (mandatory 1:1 into) mapping from domain values of a given type to data values (e.g. Figure 1(b)), so “ $x \approx y$ ” is read “ x is represented by y ”. In ORM, transformations between domain and data values are handled implicitly.

ER, UML and OWL do not distinguish between domain values and data values, so when transforming between ORM and these other languages this distinction is typically ignored. Moreover, while ORM allows values to appear in the subject role of a fact (e.g. the countrycode ‘CH’ is based on Latin), the other languages do not allow this. When mapping such facts from ORM to these languages one typically pretends that the value is an entity. In ER and UML, if the value was originally modeled as populating an attribute then one also has to remodel the attribute as an entity or relationship (e.g. consider the optional fact type PersonTitle determines Gender [13]).

Moving on from value reference, the rest of this section focuses on simple reference schemes to identify an entity either by means of a single attribute or relationship to another object, or by an individual constant. Figure 2(a) models some ways to refer to a country using ORM. Let’s agree to use ISO 3166 alpha-2 codes (e.g. “AU” for Australia, and “US” for the United States of America) for country codes, and ISO names for country names. The reference mode ““(code)” indicates that countries are

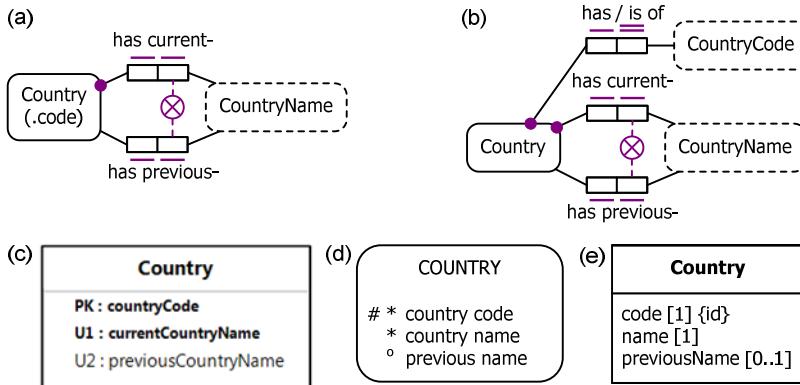


Fig. 2. A schema in (a) ORM, (b) ORM, (c) Relational, (d) Barker ER, and (e) UML notation

primarily identified by their country code, so each country has exactly one country code (for this model we ignore other country codes such as ISO 3166 alpha-3 codes) and each country code refers to at most one country.

The constraints on the current name relationship indicate that it is mandatory (each country has a current country name) and is one-to-one (each country has at most one current country name, and vice versa). The fact type **Country** has previous- **CountryName** is also one-to-one, but is *optional* for **Country**. Only a few countries have a previous name (e.g. Sri Lanka was called “Ceylon” until 1972, and Myanmar formerly had the ISO country name “Burma”). The circled “X” denotes the exclusion constraint that no country name is both a current and a previous ISO country name. This constraint is ignored in the other modeling notations.

The **Country(.code)** reference mode notation is a convenient abbreviation for the mandatory 1:1 fact type **Country** has **CountryCode** shown in Figure 2(b), where the uniqueness constraint on the role of **CountryCode** is chosen for the preferred or primary identifier (as indicated by a double bar). In ORM, the simplest kind of identification scheme for an entity type is a mandatory, 1:1 relationship from the entity type to a value type. In this example, each of the country code and current country name relationships provide such a simple identification scheme for **Country**. The previous country name relationship is not an identification scheme for **Country**, since not all instances of **Country** have to participate in this relationship. Nevertheless, previous country names are identifiers for those countries that have a previous name.

Figure 2(c) displays the relational database schema diagram generated from the ORM schema using the NORMA tool [6], an open source plug-in to Microsoft Visual Studio. The table schema for **Country** has three columns (or attributes). The **countryCode** and **currentCountryName** attributes are in bold type, indicating that they are mandatory (not nullable). The **previousCountryName** attribute is unbolded, so is optional (nullable). The “PK” marks **countryCode** as a primary key component for **Country**. Since no other columns in **Country** are marked “PK”, **countryCode** is the whole primary key of that table. The “U1” on **currentCountryName** marks it as a component of another uniqueness constraint (named **U1** in the scope of this table). No other columns in the table are marked “U1”, so entries

in the currentCountryName column are unique (no duplicates). Since currentCountryName is also mandatory, it is an alternate key for Country. The “U2” applied just to the previousCountryName column indicates that its non-null entries are unique, so those countries with a previous name can also be identified by that previous name.

Suppose we populate the Country table in Figure 2(c) with the tuple ('MM', 'Myanmar', 'Burma'). For the facts intended by this row entry, most humans would accept fact verbalizations such as: The country that has the country code 'MM' has the current country name 'Myanmar'; The country that has the country code 'MM' has the previous country name 'Burma'. These verbalizations use definite descriptions ("The country that has the country code 'MM'", "the country code 'MM'", etc.) to refer to objects. Various ways to formalize definite descriptions and address related problems (e.g. how to make sense of statements such as "The country named 'Middle Earth' does not exist") have been debated by logicians since the time of Bertrand Russell [24] till the present day (e.g. Garson's System !S [9]). Like Russell's theory of descriptions, ORM rewrites such definite expressions in terms of classical predicate logic, but it differs by formalizing the underlying reference schemes, and then completing the fact instances by adding simple existential assertions within the context of the reference scheme [10, 16]. A similar approach may be used to formalize definite descriptions within other data modeling approaches.

Figure 2(d) depicts the example schema in Barker ER notation [1], arguably the best of the popular, industrial ER versions. The “#” on the country code attribute marks it as a component of the primary identifier for the Country entity type. In this case, it is the whole primary identifier. An asterisk “*” before an attribute indicates that it is mandatory for its entity type, and a small “o” indicates the attribute is optional. Graphically, the only identifier that Barker ER can depict is the primary identifier, so it cannot mark the country name attribute as an alternate key. Nor can it display the uniqueness constraint on the non-null entries for the previous name attribute.

Figure 2(e) depicts the example as a UML class diagram. Attribute multiplicities appear in square brackets after the attribute name. A multiplicity of 1 indicates that the attribute is mandatory and single valued, and a multiplicity of “0..1” means the attribute is optional and single-valued, so each country has exactly one code, exactly one (current) name, and at most one previous name. In 2011, UML version 2.4.1 introduced the capability to declare a class property (attribute or association end owned by the class) to be a component of a value-based identifier for the class. On a class diagram, the class property is annotated with an {id} modifier. The semantics of this feature remains the same in the latest version at the time of writing, UML 2.5 beta 1 [23]. In Figure 2(e) only the code attribute is marked with {id} so this declares it as the natural identifier for the Country class. While UML does not require value-based identifiers to be declared, for human communication purposes such identifiers are essential.

UML allows at most one identification scheme per class to be depicted with the {id} modifier, so cannot graphically depict the name attribute as an alternate identifier for Country. Even if Country had no other identification schemes, we cannot use the {id} modifier to depict the previousName attribute as unique for its non-null entries, since UML does not allow the components of an {id} scheme to all be optional.

In OWL models, entities (in the sense given earlier) are typically identified by Internationalized Resource Identifiers (IRIs), such as `www.eg.org#Czech_Republic`, which function like individual constants in logic. Many country names contain embedded spaces so can't be used directly as IRIs. Use of IRIs to identify all countries of interest is not reflected in the Figure 2 data models, which are designed for database applications where country codes are standardly used to identify countries.

Some OWL ontologies use meaningless, automatically generated identifiers as IRIs just as some relational database designers prefer artificial, surrogate keys for their table schemes. Human-readable labels can be associated with IRIs using `rdfs:label` annotation properties, and many OWL software tools can be configured to display labels instead of IRIs to make ontologies more readable. Allowing that countries may be identified by IRIs, the example may be coded in OWL basically as shown below, using Manchester syntax.

```

DataProperty: hasCountryCode
  Domain: Country
  Range: xsd:string
  Characteristics: Functional
DataProperty: hasCurrentCountryName
  Domain: Country
  Range: xsd:string
  Characteristics: Functional
DataProperty: hasPreviousCountryName
  Domain: Country
  Range: xsd:string
  Characteristics: Functional
Class: Country
  SubClassOf: hasCountryCode min 1
  HasKey: hasCountryCode
  SubClassOf: hasCurrentCountryName min 1
  HasKey: hasCurrentCountryName
  HasKey: hasPreviousCountryName

```

In OWL, declaring a predicate as a *HasKey* property is similar to saying that it is inverse functional. For example, “`HasKey: hasCountryCode`” declares that each character string is the country code of at most one named country. The *HasKey* feature has to be used for this declaration, since OWL does not allow data properties (that relate individuals to literals) to be directly declared as inverse functional properties.

However, OWL does allow object properties (that relate individuals to individuals) to be directly declared inverse functional. In OWL an individual may be named (identified by an IRI) or unnamed (represented by a blank node). This allows OWL to capture *identification schemes that identify individuals by relating them directly to other individuals*. For example, the schemas in Figure 3 identify top politicians (e.g. presidents or prime ministers) by the country that they head (as chief politician).

In the ORM schema of Figure 3(a), this is modeled using the mandatory 1:1 fact type `TopPolitician heads Country`, with the double-bar indicating the uniqueness constraint for the preferred identifier. Names for the roles played by instances of `Country` appear

in square brackets beside their role. Figure 3(b) shows the relational schema generated from the ORM schema using the NORMA tool, and Figure 3(c) includes a sample fact population for this table. Here the countryHeaded primary key is used to identify the top politicians. By entering the tuple ('AU', 'GB') we can record the fact that the top politician of the country with code 'AU' was born in the country with code 'GB'. without knowing the names of that politician (Julia Gillard) and those countries (Australia and the United Kingdom).

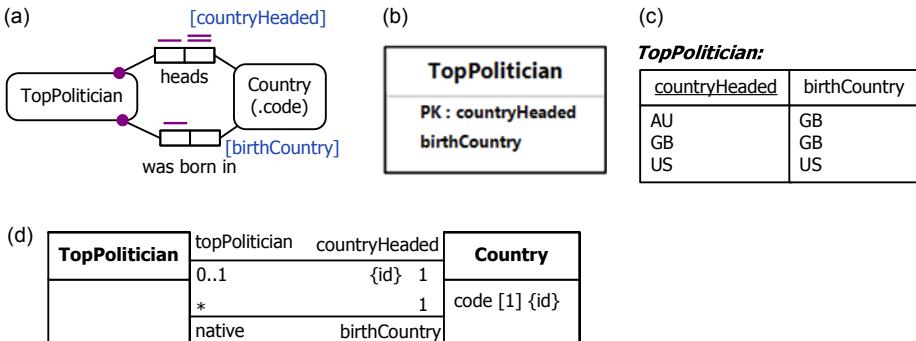


Fig. 3. Identifying top politicians by country: (a) ORM; (b), (c) Relational; (d) UML notation

Figure 3(d) shows a UML class diagram for this example, using the **{id}** modifier to mark the association end **TopPolitician.countryHeaded** as identifying instances of the **TopPolitician** class. Although Barker ER allows relationships as components of a composite primary identifier, it does not cater for cases where a single relationship provides the whole identifier [4, p. 3-13], so does not support this kind of reference scheme.

The schema may be coded in OWL using Manchester syntax as shown below. The identification scheme for **TopPolitician** is captured by specifying the object property **headsCountry** as an injective relationship from **TopPolitician** to **Country** by declaring its **minCardinality** to be 1 and characterizing the predicate as both functional and inverse functional.

```

DataProperty: hasCountryCode (see earlier code sample for details)
Class: Country
SubClassOf: hasCountryCode min 1
HasKey: hasCountryCode
ObjectProperty: headsCountry
Domain: TopPolitician
Range: Country
Characteristics: Functional, InverseFunctional
ObjectProperty: wasBornInCountry
Domain: TopPolitician
Range: Country
Characteristics: Functional
Class: TopPolitician
SubClassOf: headsCountry min 1
SubClassOf: wasBornInCountry min 1

```

The fact corresponding to the ('AU', 'GB') relational tuple may be coded by using blank nodes (represented by Skolem constants starting with an underscore) for the politician and countries. In Manchester syntax this may be declared thus:

```

Individual: _:p1
Facts: headsCountry _:c1, wasBornInCountry _:c2
Individual: _:c1
Facts: hasCountryCode "AU"
Individual: _:c2
Facts: hasCountryCode "GB"
```

Although the OWL code discussed emulates the data models in Figure 2 and Figure 3, the semantics are not precisely the same. Constraints specified in data models are not translated into actual constraints in OWL, but rather logical propositions that are understood in terms of open world semantics where some individuals may be unnamed [21]. For example, declaring the subclass restriction “hasCountryCode min 1” for Country simply says that each country has a country code—it doesn’t require the system to know what that code is. So OWL’s support for unnamed individuals differs from what might be intuitively expected by users of typical database systems.

The subjects of OWL object properties and data properties may be named individuals (i.e. individuals that are explicitly named by an IRI in the Abox (the set of fact assertions)) or anonymous individuals (denoted by blank node ids). They may also be other things (e.g. classes or properties) but such possibilities are not relevant to the topic of this paper. For discussion purposes, Figure 4 expands the ORM schema in Figure 3(a) by explicitly depicting IRI reference relationships and the countrycode relationship. As the constraints indicate, each IRI identifies at most one entity (in this case, a top politician or a country), but any given entity may have more than one IRI (by default, OWL does not apply the Unique Name Assumption). For example, if we add the following facts to the previous OWL schema, an OWL reasoner will now correctly infer that Myanmar and Burma are the same country.

```

Individual: Myanmar
Facts: hasCountryCode "MM"
Individual: Burma
Facts: hasCountryCode "MM"
```

In OWL, an *InverseFunctional* characteristic may be applied only to an object property, whereas both object properties and data properties can be declared as *HasKey* properties. Moreover, the uniqueness aspect of *HasKey applies only to named individuals*. In contrast, *InverseFunctional* declarations apply to any kind of individual (named individuals, anonymous individuals, and individuals whose existence is implied by existential quantification) [29, sec. 9.5].

Given our current OWL schema for top politicians, we can assert that Julia Gillard was born in some country that has country code “GB” as follows, without knowing which country has that country code.

```

Individual: JuliaGillard
Facts: wasBornInCountry _:c1
Individual: _:c1
Facts: hasCountryCode "GB"
```

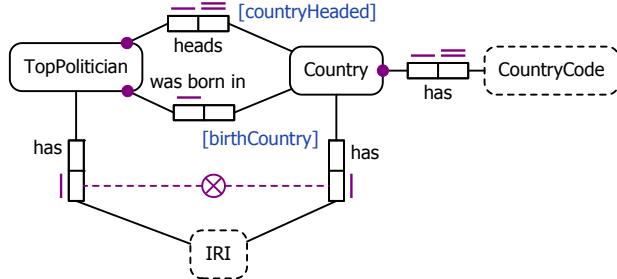


Fig. 4. A simplified view of adding IRI references to the ORM schema in Figure 3(a)

Now suppose we discover that the United Kingdom has the country code “GB”, and we choose “TheUK” (in the scope of the current document) as an IRI for the United Kingdom. We can now add the following code to assert that the UK has the country code “GB”.

```

Individual: TheUK
Facts: hasCountryCode "GB"

```

One might now expect that it is correct to infer the following fact that Julia Gillard was born in the United Kingdom:

```

Individual: JuliaGillard
Facts: wasBornInCountry TheUK

```

However, this fact does not follow! HasKey declarations apply only to named individuals (with IRIs explicitly asserted). So declaring hasCountryCode as a HasKey property for Country says that at most one *named* individual has a given country code, while allowing that there could be many unnamed individuals with that same country code. Our assertion that Julia Gillard was born in some (i.e. at least one) country (which might be named or unnamed) that has the country code “GB” does not imply that this is the same as the named country (TheUK) that has that country code. This restriction of HasKey semantics to named individuals is designed to help ensure that inference rules using HasKey properties are “DL-safe” [27, sec. 9.5], so they will execute in a finite time. The choice of the OWL term “HasKey” is perhaps unfortunate, as this notion is weaker than key declarations in data modeling approaches.

Since typical database users often assume closed world semantics for most of their data, and expect constraints to be enforced accordingly, considerable care is required when transforming to or from OWL ontologies that typically adopt open world semantics and use logical conditionals instead of constraints. To address this issue, some recent proposals have been made to extend OWL with the ability to classify Tbox propositions as either derivation rules or as constraints [22], or to classify selected types and properties as complete by making them DBox predicates [7].

3 Compound Reference Schemes

This section considers *compound reference schemes*, where an entity may be identified by means of a combination of two or more attributes or relationships. Figure 5 displays some examples in the four data modeling notations under consideration.

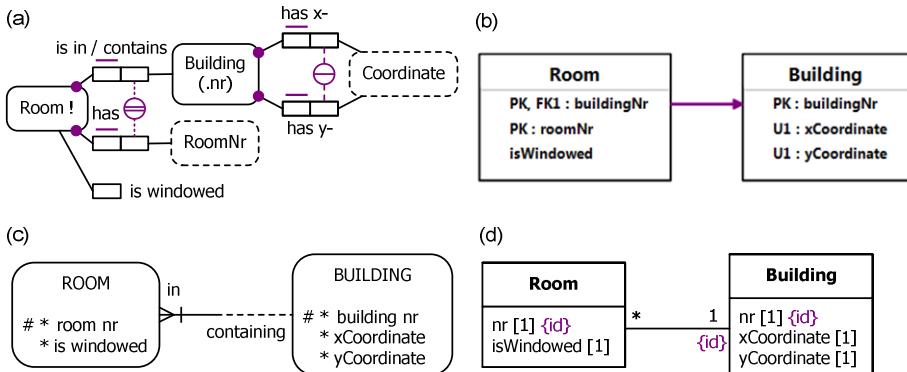


Fig. 5. Compound reference in (a) ORM, (b) Relational, (c) Barker ER, (d) UML notation

In the ORM schema of Figure 5(a), the circled double bar denotes an external uniqueness constraint underlying a compound, preferred reference scheme for Room (each room is identified by combining its building with its local room number). This corresponds to fact verbalizations that identify rooms by compound definite descriptions (e.g. the Room that has RoomNr 205 and is in the Building with BuildingNr 3). The circled single bar depicts an external uniqueness constraint for a compound, alternate reference scheme for buildings based on a combination of their x and y coordinates. Buildings also have a simple, preferred reference scheme (based on building number). The unary fact type Room is windowed records which rooms have a window.

Figure 5(b) shows the relational schema generated by NORMA from the ORM schema. The preferred reference schemes for Room and Building are indicated by their primary key attributes, and the alternate reference scheme for Building is captured by marking the coordinate pair as unique and mandatory. To save space, display of data types is suppressed (the isWindowed attribute is Boolean).

Figure 5(c) shows a Barker ER schema for the same example. The composite reference scheme for Room is indicated by the # on the room nr attribute and the vertical stroke “l” through Room’s role in its containment relationship with Building. The simple reference scheme for Building is captured by the # on the building nr attribute, but Barker ER has no way to indicate that the building coordinate pair provides an alternate reference scheme for Building.

Figure 5(d) shows a UML class diagram for the same example. The composite reference scheme for Room is captured by marking the attribute Room.nr and the association end Room.building with the {id} qualifier. The simple reference scheme for

Building is indicated by the {id} modifier on Building.nr, but UML cannot graphically depict that the coordinate combination is also unique for buildings.

In OWL, most of the example may be coded in a similar way to that discussed earlier. The isWindowed predicate may be declared as a data property with domain Room and range xsd:Boolean, with individual facts using true or false as appropriate. For the external uniqueness constraints in Figure 5(a), composite HasKey properties are declared as shown below assuming the relevant properties are defined.

```

Class: Building
HasKey: hasBuildingNr
HasKey: hasXcoordinate, hasYcoordinate
Class: Room
HasKey: isInBuilding, hasRoomNr

```

However, these HasKey declarations have no effect on specific room or building instances unless an IRI is also explicitly declared for those instances. In this case, one could invent meaningful IRIs (e.g. Room3-205 for room 205 in building 3), but in cases where this is impractical (e.g. consider IRIs for street addresses), one might use surrogate identifiers (e.g. address_1, address_2, etc.) or instead simply abandon any attempt to capture the uniqueness semantics in the OWL ontology.

4 Disjunctive and Context-Dependent Reference Schemes

In a *disjunctive reference scheme*, instances of an entity type are identified by a logical disjunction of attributes or relationships, at least one of which is optional for the entity type, while the disjunction itself is mandatory for the entity type. Such schemes violate entity integrity (where all primary key components must be non-null), so cannot be implemented as primary keys in a pure relational model, but may be implemented in SQL systems since they do not require primary keys to be declared. To our knowledge, disjunctive reference was first investigated at the relational level by Thalheim [25]. Our initial research on disjunctive reference at the conceptual level is discussed in [19]. The treatment in this section introduces extensions to our earlier work.

Figure 6(a) shows a tiny fragment of the ORM metamodel. The external uniqueness constraint ensures that if a role has a name then that name is unique within its predicate. The constraint has *inner join semantics*, so the relation scheme Role(roleid, predicateid, [rolename]) formed by inner-joining the two relation schemes has a uniqueness constraint over the attribute-pair (predicateid, roleName). This allows many unnamed roles within an ORM predicate.

The external uniqueness constraint shape in Figure 6(b) has an added “o”, indicating *outer join semantics*. The relation scheme Course(courseCode, courseTitle, [departmentCode]) formed by outer-joining the two relation schemes has a uniqueness constraint over the attribute-pair (courseTitle, departmentCode), with the added proviso that nulls are treated as actual values for this constraint. So the same course title may apply to at most one course with no department, but may also apply to many courses in different departments.

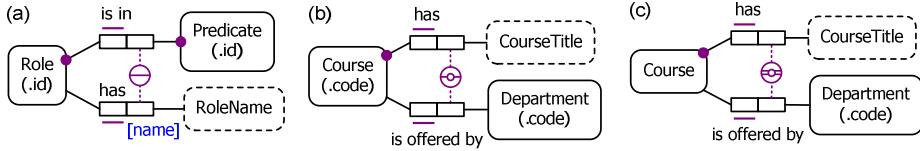


Fig. 6. Disjunctive reference with (a) inner join semantics; (b), (c) outer join semantics

The schema in Figure 6(c) also has outer join semantics, but the double-bar in the external uniqueness constraint shape indicates that this is used for the preferred reference scheme for Course (course codes are not used). When the uniqueness constraint involves an optional relationship, inner join semantics can't be used for the preferred reference scheme. For example, if we remove the RoleId reference scheme from Figure 6(a), then a predicate with two unnamed roles has no way to identify its roles.

Figure 7 shows the weakest pattern that can be used in ORM to reference all instances of an entity type A . This involves a set of n functional binary relationships ($n > 1$) whose first roles are disjunctively mandatory (depicted by the circled dot) and whose second roles are restricted by an external uniqueness constraint with outer join semantics. A formalization of this pattern (using a superseded graphical notation) is provided in [19]. In practice, additional constraints often apply that allow more compact formalizations. For example, the external uniqueness constraint in Figure 6(b) may be formalized as follows, where the first conjunct captures the inner join semantics and the second conjunct captures the additional semantics for the outer join:

$$\begin{aligned} \forall ct:\text{CourseTitle}, d:\text{Department} \exists^{0..1} c:\text{Course} (c \text{ hasCourseTitle } ct \& c \text{ isOfferedBy } d) \\ \& \forall ct:\text{CourseTitle} \exists^{0..1} c:\text{Course} [c \text{ hasCourseTitle } ct \& \sim \exists d:\text{Department} (c \text{ isOfferedBy } d)] \end{aligned}$$

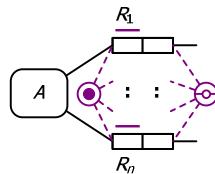


Fig. 7. Weakest pattern for disjunctive reference

Our original work on disjunctive reference was motivated by the need to handle such cases in industrial modeling, such as botanical classification schemes or complex addressing schemes. A simplified treatment of the botanical case is discussed in [18, pp. 522-523], along with options for mapping it to a relational database. The full treatment of the botanical classification model in [24] illustrates just how complex disjunctive reference schemes can be in practice.

Figure 8 shows two further cases of disjunctive reference. These use a new construct we just added to ORM. A uniqueness bar with a dotted line over it indicates that this provides a way to *reference just some instances* of the relevant entity type.

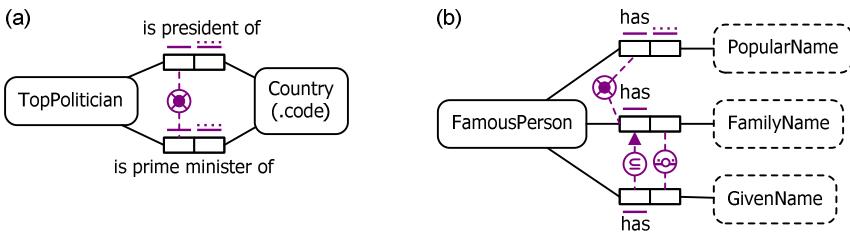


Fig. 8. Further cases of disjunctive reference with (a) inner join and (b) outer join semantics

In Figure 8(a), a top politician is identified by the country of which he/she is president or prime minister. The exclusive-or constraint ensures exactly just one of these relationships applies to any given top politician. Figure 8(b) models a situation where some famous persons may be identified by their popular name (e.g. ‘Confucius’ instead of ‘Kong Qiu’), others simply by their family name (e.g. ‘Einstein’ and ‘Newton’ denote Albert Einstein and Isaac Newton), while others may be identified by a combination of their family name and a specific given name (e.g. Marie Curie, Pierre Curie). The circled subsethood operator depicts the subset constraint that each famous person with a given name also has a family name.

There is no space here to discuss in detail how disjunctive reference may be implemented in relational database, but in general it is usually best to introduce a simple identifier for the primary key, moving the disjunctive reference to an alternate key. The simple identifier can be either a surrogate or a meaningful identifier obtained by using a derivation rule to concatenate the components (type casting data types to string where needed). For some basic discussion of mapping alternatives see [18].

Disjunctive reference is not supported in the graphical notation of Barker ER or UML. In OWL, HasKey properties have inner join semantics, so cases like that of Figure 6(a) can be coded, along with the usual limitations discussed earlier for HasKey properties. Basic work on mapping ORM to OWL or description logic has been carried out by several researchers (e.g. [20, 8]). We leave it as a research question to explore to what extent OWL can deal with the other cases of disjunctive reference discussed in this section

In a *context-dependent reference scheme*, the preferred identifier for an entity varies according to the context. ORM supports this notion by allowing subtypes to introduce new preferred reference schemes used within the scope of their immediate fact types. For example, in the ORM schema shown in Figure 9 student employees have a global person number, but are identified by their student number for degree enrolment facts, and by their employee number for employment and tutoring facts (a solid subtyping arrow implies inheritance of preferred reference scheme). For further discussion of this topic, along with relational mapping options, see [18, pp. 519-521]. Barker ER and UML have no direct support for this notion, but UML’s implicit use of oids to identify members of classes provides built-in support for global identifiers. While OWL does not formalize the notion of preferred reference, its allowance of multiple IRIs for the same entity and use of the owl:sameAs predicate to equate individuals provides one way to emulate such context-dependent reference.

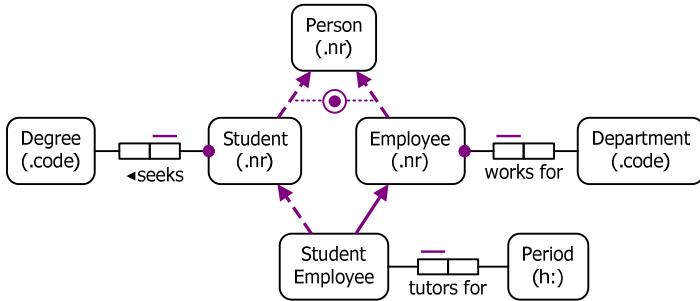


Fig. 9. An ORM schema with context-dependent reference schemes

5 Conclusion

Popular languages for data modeling and ontology modeling differ substantially in their support for reference schemes, so an understanding these differences is important for transforming models from one language to another. This paper provided a comparative review of how reference schemes may be modeled in ORM, Relational databases, Barker ER, UML and OWL.

While all these languages provide basic support for some simple and compound reference scheme patterns, various limitations in the latter three languages were identified with respect to their support for other reference scheme patterns. For example, Barker ER and UML do not graphically support secondary reference schemes, Barker ER does not support simply entity-to-entity reference, Barker ER and UML have no intrinsic support for disjunctive reference, and OWL's use of HasKey properties is restricted to inner join semantics and named individuals.

Some new cases of disjunctive reference were illustrated, and new ORM constraint symbols were introduced to distinguish the ways in which internal and external uniqueness constraints underlie different kinds of reference schemes. While these symbols have now been adopted in ORM, we have yet to fully support them in the NORMA tool (e.g. with distinguished automated verbalizations and relational mapping options). Such implementation support is planned for the near future.

Further research and development work is encouraged to extend support for reference schemes in the various languages, as well as datalog [1, 17], and to provide automated transformation of models between them.

Acknowledgements. The new uniqueness constraint symbols in section 4 arose from collaborative discussion with Matthew Curnand. We are also grateful to Ed Seidewitz for clarifying basic semantics of the {id} modifier recently introduced to UML.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Allen, J.: Natural Language Understanding, 2nd edn. Benjamin/Cummings, Redwood City (1995)
3. Bakema, G., Zwart, J., van der Lek, H.: Fully Communication Oriented Information Modelling, Ten Hagen Stam (2000)
4. Barker, R.: CASE*Method: Entity Relationship Modelling. Addison-Wesley, Wokingham (1990)
5. Chen, P.P.: The entity-relationship model—towards a unified view of data. ACM Transactions on Database Systems 1(1), 9–36 (1976), <http://csc.1su.edu/news/erd.pdf>
6. Curland, M., Halpin, T.: The NORMA Software Tool for ORM 2. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBP, vol. 72, pp. 190–204. Springer, Heidelberg (2011)
7. Patel-Schneider, P.F., Franconi, E.: Ontology Constraints in Incomplete and Complete Data. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 444–459. Springer, Heidelberg (2012)
8. Franconi, E., Mosca, A., Solomakhin, D.: ORM2: Formalisation and Encoding in OWL2. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM-WS 2012. LNCS, vol. 7567, pp. 368–378. Springer, Heidelberg (2012)
9. Garson, J.: Modal Logic for Philosophers. Cambridge University Press, Cambridge (2006)
10. Halpin, T.: A Logical Analysis of Information Systems: static aspects of the data-oriented perspective. Doctoral dissertation, University of Queensland (1989), http://www.orm.net/Halpin_PhDthesis.pdf
11. Halpin, T.: ORM 2. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 676–687. Springer, Heidelberg (2005)
12. Halpin, T.: Fact-Oriented Modeling: Past, Present and Future. In: Krogstie, J., et al. (eds.) Conceptual Modelling in Information Systems Engineering, pp. 19–38. Springer, Berlin (2007)
13. Halpin, T.: Object-Role Modeling: Principles and Benefits. International Journal of Information Systems Modeling and Design 1(1), 32–54 (2010)
14. Halpin, T.: Structural Aspects of Data Modeling Languages. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBP, vol. 81, pp. 428–442. Springer, Heidelberg (2011)
15. Halpin, T.: Fact-Orientation and Conceptual Logic. In: Proc. 15th International EDOC Conference, pp. 14–19. IEEE Computer Society, Helsinki (2011)
16. Halpin, T.: Formalization of ORM Revisited. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM-WS 2012. LNCS, vol. 7567, pp. 348–357. Springer, Heidelberg (2012)
17. Halpin, T., Curland, M., Stirewalt, K., Viswanath, N., McGill, M., Beck, S.: Mapping ORM to Datalog: An Overview. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6428, pp. 504–513. Springer, Heidelberg (2010)
18. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)
19. Halpin, T., Ritson, R.: Fact-Oriented Modelling and Null Values. In: Srinivasan, B., Zelezniakov, J. (eds.) Research and Practical Issues in Databases. World Scientific, Singapore (1992)

20. Keet, C.: Prospects for and issues with mapping the Object-Role Modeling language into DL RIfd. In: 20th Int. Workshop on Description Logics (DL 2007). CEUR-WS, vol. 250, pp. 331–338 (2007)
21. Krötzsch, M., Simancik, F., Horrocks, I.: A Description Logic Primer, eprint arXiv:1201.4089 (2012), <http://arxiv.org/abs/1201.4089>
22. Motik, B., Horrocks, I., Sattler, U.: Bridging the Gap Between OWL and Relational Databases. *J. of Web Semantics* 7(2), 74–89 (2009)
23. Object Management Group: OMG Unified Modeling Language (OMG UML), version 2.5 FTF Beta 1 (2012), <http://www.omg.org/spec/UML/2.5>
24. Ritson, P.: Use of Conceptual Schemas for Relational Implementation. Doctoral dissertation, University of Queensland (1994)
25. Russell, B.: On Denoting. *Mind* 14, 479–493 (1905)
26. Thalheim, B.: On Semantic Issues Connected with Keys in Relational Databases. *J. Inf. Process. Cybern.* 1(2), 11–20 (1989)
27. W3C: OWL 2 Web Ontology Language: Primer, 2nd edn. (2012),
<http://www.w3.org/TR/owl2-primer/>
28. W3C: OWL 2 Web Ontology Language: Direct Semantics, 2nd edn. (2012),
<http://www.w3.org/TR/owl2-direct-semantics/>
29. W3C: OWL 2 Web Ontology Language Manchester Syntax, 2nd edn. (2012),
<http://www.w3.org/TR/owl2-manchester-syntax/>
30. W3C: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, 2nd edn. (2012), <http://www.w3.org/TR/owl2-syntax/>
31. Wintraecken, J.: The NIAM Information Analysis Method: Theory and Practice. Kluwer, Deventer (1990)

Visually Capturing Usage Context in BPMN by Small Adaptations of Diagram Notation

Guttorm Sindre, John Krogstie, and Sundar Gopalakrishnan

Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
Sem Sælands Vei 7-9, 7491 Trondheim, Norway
`{guttors,krogstie,sundar}@idi.ntnu.no`

Abstract. For mobile and multi-channel information systems it is often relevant to model where something is supposed to take place and what equipment is used. Whereas traditional business process modelling notations seldom capture context, this paper looks at the possibility of extending BPMN diagrams with visual means for capturing location and equipment types used in mobile work processes. A wide range of possible ways for doing this visually is discussed and illustrated, whereupon the most viable alternatives are compared analytically mainly based on principles suggested by Moody. The results indicate that there are several approaches to include such usage context information visually with reasonably good results, but that approaches using intuitive icons to reflect locations and equipment types appear to be the most promising.

1 Introduction

We observe that context of work e.g. 'where' (i.e. location) something is done, and with what equipment is getting increasingly important in mobile and multi-channel information systems. It is also possible to utilize context to a larger degree (also real time) to know where users, equipment and goods should be, or where and when to provide a certain service through different devices providing a flexible, multi-channel work process support. For mobile and multi-channel information systems it is often relevant to model both the location of the user and the equipment with which the user intends to access the system.

Even if geographical location and equipment is included in some enterprise architecture frameworks, business process modelling notations seldom capture such aspects. High-level process notations such as Use Case diagrams [1] and textual use case descriptions [2] normally do not include location or equipment, and the same applies to more detailed workflow representations such as BPMN and UML activity diagrams. As an example, consider the BPMN diagram for a part of the work process executed by a municipal home care unit. The process starts with the Shift Leader assigning Home Care Assistants (HCA) to the clients that the unit is responsible for, typically elderly people who can still live in their homes if they receive some help with cleaning and hygiene, shopping, cooking, etc. There are multiple HCAs,

indicated by the three parallel lines symbol at bottom of the large "Make daily round" activity, meaning that all the HCAs make their daily rounds in parallel here, to the clients assigned to them. The HCA first decides on the visit sequence, the latter supported by some software application calculating the optimal driving route, but possibly considering some constraints wrt. certain clients needing to be visited in certain time intervals. Then each client on the HCA's list is visited, in the "Visit Client" activity, which is looped until all clients on the list have been visited, cf. the "Done" arrow going right from the X diamond. During the day, the Shift Leader reviews the progress of the HCA's, being informed if any of them experience delays or problems (e.g., one client needing more attention than usual, or heavy traffic or accidents delaying the HCA). If a HCA is so much delayed that he can no longer make it through his list of patients in the allotted time, the Shift Leader may reassign some patients to other HCA's, who then need to make a new decision on their visit sequence. For each visit, the HCA goes through a preparation task of finding relevant information about the client. In most cases only limited preparation is needed because the HCA knows the client well from before, but in case of reassessments due to delays or replacements due to sick leaves, more preparation may be needed, such getting to know about special needs of the client, health problems that the HCA must be aware of, medication that the client needs to take, and whether the HCA should help the client take the medication or if it is enough just to check or ask the client if it has been remembered. At the client's the HCA then does the necessary house work, and also orders goods such as groceries, hygiene articles, light bulbs etc in addition to procuring additional medicine if necessary. For simplicity the model does not show what happens further to this order. Finally, information about the visit is registered in the "Log visit" task and the HCA moves on to the next client until done with all clients for the day.

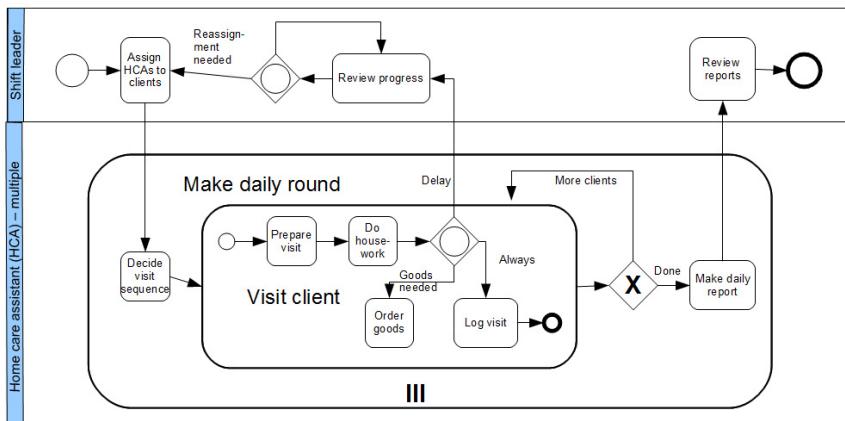


Fig. 1. BPMN example, part of a work process in home care

This diagram tells nothing about mobile aspects of the work process, although one might guess that there is necessarily some movement involved, i.e., the HCA needs to get to the homes of the various clients, and the "Do housework" task must definitely be performed there. However, for all the other tasks there are several options. For instance, the "Prepare visit" task can be performed in an office before going out to the client, in the car while travelling to the client, in the parked car before entering the client's home, or even inside the client's home. Similarly, the "Log visit" task can be performed in a number of different locations. Also, the tasks can be performed with a wide variety of equipment. Both locations and equipment may have a lot of impact on requirements for applications supporting these tasks. In some previous papers we have investigated how to include location in UML Activity Diagrams [3-6], how to include location in use case diagrams [7], and how to include equipment information in use case diagrams [8]. The current paper provides the following extensions to these previous works:

- Using BPMN rather than UML activity diagrams, this because BPMN seems is the preferred notation for business process modelling in many settings.
- Considering diagram notations that will allow the capture of *both location and equipment* at the same time. Previously, we looked at only one extension at a time. Looking at several conceptual extensions at the same time is more complicated, since a visual variable which is used for one purpose cannot also be used for another purpose.

Otherwise, the results from the previous papers are clearly relevant, and can be built upon. Some notations for including location in UML Activity Diagrams were compared analytically in [3, 6] and some of the alternatives that did best in the analytical evaluations were compared experimentally to a "zero" alternative using standard UML notes in [4, 5].

The research questions for this paper are as follows:

- **RQ1:** How can one show several usage context variables (e.g., location, equipment, ...) in the same business process diagrams, using existing BPMN notation with small adaptations?
- **RQ2:** How can the usage of visual variables and text best be combined to balance the complexity of the diagram?
- **RQ3:** What additional requirements would this put to the functionality of the modelling tool?

The focus on small adaptations (RQ1) is motivated by that there is a lot of investment in existing notations, in terms of developer training and tool implementations. Suggesting a radically different notation would disregard this investment, and although the new notation might be theoretically optimal for mobile work processes, the likelihood of industrial adoption would be smaller.

The rest of the paper is structured as follows: Section 2 presents related work then section 3 discusses the research method, i.e. framework used for the analytical evaluation. Section 4 maps out the solution space in terms of viable notation alternatives, and explain why some alternatives are obviously poor and are therefore excluded from further analysis. Section 5 presents the results of the analytical evaluation, whereupon section 6 discusses the findings and concludes the paper.

2 Related Work

Mobile applications need to take context into account to deliver services that are appropriate for their users. [9] provides several different context-dimensions that can be of interest. The spatio-temporal context describes aspects related to time and space. The environmental context captures the entities that surround the user, for example, physical objects, temperature, light, humidity and noise. The personal context describes the user state, such as pulse, blood pressure, and body temperature. The task context describes what the user is doing and other related tasks. The information context entails what information is available at a given time. Finally, the social context describes information about other actors e.g. friends, neighbours, co-workers, and relatives. As for spatial mobility, [10] distinguish between travelling, visiting and wandering. [11] adds mobile work proper, i.e., work that by nature entails moving about. Work on mobile ontologies by Veijalainen [12] supports the idea of the ‘where’ aspect as essential in mobile processes. [13,14] distinguish between “space” and “place”, the former describes geometrical arrangements that might structure, constrain, and enable certain forms of movement and interaction; “place” denotes the ways in which settings acquire recognizable and persistent social meaning in the course of interaction. In this work we look at the modelling of ‘place’. Work combining modelling of space and conceptual aspects are described in e.g. [18]. When modelling ‘place’, a limited number of places are relevant in most cases.

There are several existing proposals for how to adapt business process diagram notations for mobile systems. [16] proposes an extension of BPMN to include location constraints for activities and actor nodes. Visually this is achieved representing locations as parallelograms which can then be linked either to single activities or groups of activities. Midway on the link, a hexagon symbol is used. Having “=” inside the hexagon means a positive constraint (activity A is to be performed in location L), while “≠” means a negative constraint (activity A should not be performed in location L). In [17] a similar proposal is made for UML activity diagrams. [18] proposes extensions to UML activity diagrams specifically targeting the modelling of mobile systems. They offer one *responsibility centred notation*, where swim-lanes are used to indicate “who” performs the various process steps and location is instead added by textual annotation, and also *location centred notation*, where swim-lanes indicate “where” an action is performed, so that “who” must instead be explained by textual annotation. Responsibility centred notation reflects the normal usage of swim-lanes in business process models, while the location centred notation is a much larger deviation from normal practice. This resembles a notation which was briefly tried in [3] using swim-lanes for “where” and instead showing “who” by connecting process steps to stick figures (as in use case diagrams), however that notation was quickly dismissed because the need for linking stick men to process nodes created a lot of extra and often crossing lines in the diagram. [19] proposes an adaptation of UML sequence diagram which uses a location centred notation similar to that of [18]. Another extension of UML sequence diagrams with a similar location centred approach is [20]. In [21] another kind of visual adaptation is used, namely adding small icons inside each business process step. This extension is not made for

the purpose of capturing user context, however, only to make ordinary business process models more understandable by introducing representational redundancy in that the icons double up for the name label of the process step in indicating the nature of the activity. In total, 25 different icons are proposed. Conceptually this is much further away from our purpose than the works mentioned in the previous paragraph; however the notational strategy of using intuitive icons might be equally relevant as using parallelograms or other more abstract visual means.

3 Research Method

Notations can be evaluated either analytically or empirically. Empirical evaluations will always be the ultimate target, since these can tell whether the modelling approaches work in practice and are accepted by prospective users. However, with a wide range of notation alternatives, there will be a combinatorial explosion of empirical comparisons to be made in experiments or case studies to evaluate all possibilities. Therefore, a more viable approach is to start by an analytical evaluation to indicate a smaller number of notation candidates which - according to the principles underlying the analytical evaluations - seem to have a good likelihood of success, and then compare these empirically to find out which is the better one. In this paper we will therefore evaluate alternatives analytically, and the key question then becomes which evaluation framework to use.

The Bunge-Wand-Weber (BWW) ontology [22] is often used for analytical evaluations of conceptual modelling languages, and was used to evaluate BPMN in [23]. However, the BWW ontology only concerns itself with the underlying conceptual basis of a modelling language, not the visual representation. A semiotic quality framework (SEQUAL) for conceptual model quality originated in [24] has later been extended and adapted for analytical evaluations of modelling languages [25], evaluations of BPMN presented in [26,27,28]. SEQUAL covers both concepts and visual representation and is in [25] extended to include the principles of good notation proposed by Moody [29], which are our basis here with its nine principles:

- 1) *semiotic clarity* means a 1:1 mapping between graphical symbols and concepts.
- 2) *perceptual discriminability*: How easily and accurately can symbols be differentiated from each other?
- 3) *semantic transparency*: How well does a symbol reflect its meaning?
- 4) *complexity management*: What constructs does the diagram notation have for supporting different levels of abstraction, information filtering, etc.?
- 5) *cognitive integration*: Does the notation provide explicit mechanisms to support navigation between different diagrams?
- 6) *visual expressiveness*: To what extent does the notation utilize the full range of visual variables available?
- 7) *dual coding*: Using text to complement graphics.
- 8) *graphic economy*: Avoiding a too large number of different symbols.
- 9) *cognitive fit*: Trying to adapt the notation to the audience

Not all of these 9 principles are equally relevant for our purposes, so we choose to discard two for our evaluation:

- Cognitive Integration investigates the relationship between several sub-languages within one bigger language. We only look at process models
- Cognitive fit concerns the usage of different dialects of a language for different stakeholders. At the current stage we are only looking at one dialect.

Our focus, as argued in the Introduction, is to use only minor adaptations of a mainstream notation. Hence, we will add one more criterion: *Minimal deviation from mainstream notation*. The detailed evaluations done in the scoring will be presented in the section with the results, since this is easier to explain when having concrete notation examples to refer to. In the next section we will discuss which notation alternatives to consider for the evaluation.

4 Notation Alternatives

The wanted extension is to include usage context, such as location, movement, and equipment used in process models expressed in BPMN. In general there are two main ways that more information can be included in a process diagram:

- making changes to nodes or arrows that are already present in the diagram
- introducing *new* graphical elements in the diagram

Both for changes to existing graphical elements and introduction of new graphical elements we have a number of visual variables at our disposal. Bertin [30] suggests 8 different variables: Two planar variables and six retinal variables. The planar variables are *horizontal* and *vertical position*, i.e., the meaning of a symbol, or its relationship to others, is partly explained by the symbol's placement in the diagram. This is already used in BPMN. A process node placed within a certain swim-lane is performed by the role responsible for that swim-lane, whereas a node outside that swim-lane is performed by somebody else. Moreover, associations are also indicated by means of placement: if an arrow is placed between two process nodes, touching at each end, then these two process steps are related, e.g., with the meaning that one precedes the other in the process. Finally, there is a common convention in process modelling that order is also indicated by placement from top to bottom or left to right, although this can be overridden by arrows.

The six retinal variables are *shape*, *size*, *orientation*, *colour*, *brightness*, *texture*. Some of these are already used in BPMN, in particular shape. Orientation is normally used for relationships, i.e., it makes a big difference whether an arrow points one way or the other. The other variables are less used. Size may sometimes be used accidentally, e.g., some nodes being bigger than others either because they are further decomposed or because a longer textual label needs to be fitted inside, but then size does not have a precise or intended meaning. Also, it is easy to find a lot of examples where colour has been used in BPMN diagrams, e.g., using different colours for different swim-lanes or different groups of process nodes, but again this seems to be mostly for illustrative purposes, not assigning any precise meaning to the various

colours. Colour could be used in a lot of different ways in a diagram, e.g., nodes could be filled with different colours to convey different meanings, the bordering lines of nodes could have different colours, text inside the nodes could have different colours, and arrows between nodes could have different colours for different kinds of relationships. Similarly, texture could be in the form of different node fills, different line types (full, dotted, dashed) on the node border, etc.

Since we are considering two extensions to the notation at the same time - adding both locations and equipment types for the process steps, there is potentially a vast number of notations that one could choose from, namely the full combinatorial explosion of all the visual tricks indicated in two bullet items just above. Moreover, it would also be possible to use redundant coding, i.e., using several visual variables together to show the same concept. All in all, this makes for hundreds of possible combinations – but most of them are no good. Hence it is important to prune our search space by excluding the obviously poor combinations before going into the more time-consuming parts of the evaluations with detailed scoring of alternatives.

4.1 Excluding Obviously Poor Alternatives

Examples of notation choices that are excluded before the detailed evaluation are shown in Figure 2.

Each row in this Figure is indicating how one particular visual variable can be used to encode some usage context information for process nodes. In the first row, the placement of the "Log visit" activity in some planar pool indicates the location where the activity is performed. In rows 2-4 we orthogonally combine two tricks on the text label inside the node, which could then encode both location and equipment type, e.g. horizontal alignment for location, vertical for equipment type (row 2), size for location, colour for equipment type (row 3), font for location, brightness for equipment type (row 4). The subsequent rows encode just one concept each, e.g., thick, medium or thin bordering line representing three different locations. We could have shown how to encode both location and equipment types by a number of orthogonal combinations of these tricks, but that would have given an unnecessarily big amount of rows in the Figure.

The planar alternative in the top row in Figure 2 was also proposed in [18] and termed a *location centred notation*. This is not all that bad, as discussed in [6] it can be quite intuitive and therefore have good semantic transparency if the planar variable is used for location – though not for equipment [8]. The problem here, however, is that we are focusing on minor adaptations of BPMN, where the standard approach is to use pools and swim-lanes for organizational responsibility, i.e. a *responsibility centred notation* [18]. Changing this would imply a quite big deviation, potentially confusing for users who have already learnt BPMN. In particular, with this approach one would have to find another way to indicate *who* performs the activity (which, by the approaches from row 2 onwards can still easily be done simply placing the activity node in the swim-lane of the Home Care Assistant). Hence, the planar alternative is discarded in this paper.

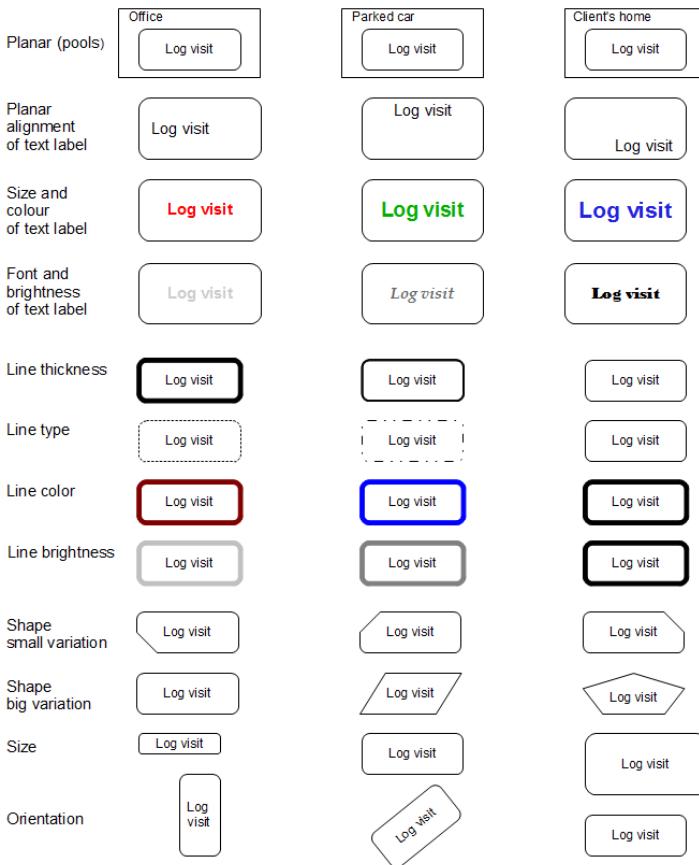


Fig. 2. Visual variables discarded from further consideration. In case of b/w printing, row 3 has red, green and blue text in the process node labels (left, middle, and right columns respectively), and row 7 has brown, blue, and black boundaries on the activity nodes.

Having thus excluded a number of visual alternatives which were obviously poor already in previous evaluations, we are left with some fewer alternatives which are more promising. In the following subsection we will describe some main notation alternatives, those which encode location and equipment with visual elements inside the existing diagrams nodes (i.e., particularly the activity nodes). Another possibility would be to rely on putting context information in new nodes. We focus on the first option here, since extra nodes easily make the overall diagram overly complex [3], specifically taking into account the large graphical complexity found in BPMN [31].

4.2 Encoding Context Information Inside the Activity Nodes

In Figure 3 we have shown alternatives only making changes internally in the activity nodes to accomplish the necessary encoding of a location and equipment type for the

activity in question. The first three rows only use the label text itself. The first row is the naive approach of putting more information into the natural language text label, which is not an extension of the language, but useful to include as a kind of zero alternative. The downside of this is that the label becomes longer and harder to comprehend. It also breaks with the normal naming convention for process steps (verb + noun). The second row uses tags to add the context information. This has several advantages, both in making it clearer for the reader what the different parts of text inside the node are, and allowing for more sophisticated tool support, for instance in showing or hiding this information on request from the user. Also, the tag approach is semantically more precise. With the plain text approach of the first row, "Log visit in patient's home" is actually ambiguous, as the phrase "in the patient's home" might simply describe where the visit took place, not necessarily where the logging activity takes place. With the tags, it could be formally defined - and explained to users - that the <loc> tag always identifies the location where the activity is to be performed. So the tagged approach of the second row has better *semiotic clarity* than the naive approach of the first row. The third row of Figure 3 illustrates the possibility of redundantly combining tags and font colours, to further enhance perceptual discriminability. However, to find out what the location is, close inspection of the text would still be needed.

The various graphical alternatives in row 4-9 of Figure 3 illustrate quite well some challenges related to achieving these two extensions (i.e., capturing both location and equipment) by modifications inside the node itself. Having seen in previous experiments [4,5] that both colour and pattern-fills did reasonably well when only one extension had to be made, it could be tempting to try to use these two orthogonally to achieve two extensions. In row 4 the location is encoded by colour, the equipment by the pattern, but this entails some problems. First of all, it is not always straightforward to combine colours and patterns. Too bright colours, e.g. yellow, will cause the pattern not to be easily visible, at least if the pattern is on a white background. Hatchings based on thin lines will combine poorly even with darker colours, as it will be hard to distinguish whether a hatching is black, blue or green. Hence, it may be necessary to use patterns that are quite dense, like the ones in Figure 3, to allow for easy distinction of the colours. Second, patterns often interfere with the text, making it hard to read. This is easily fixed using a white rectangle just around the label. However, if there is a lot of text inside the label, this would give reduced space for the pattern and colour, again making perceptual discriminability poorer. Third, being limited to dense patterns rather than sparse ones means that we are able to distinguish fewer different alternatives (in this example equipment types, but in general whatever the patterns are used to capture) before the patterns become so similar to each other that they are hard to discriminate reliably, maybe only 3-4.

Seeing the problems with pattern fills, rows 5 and 6 of Figure 3 instead try to combine colour with brightness, still using red for client's home and blue for car, but now using darker for PDA and lighter for laptop. This seems to work slightly better than the colour + pattern combination, especially in terms of less interference with the text label. Black text will be harder to read the darker the background colour, but at some stage one could invert the text label to white instead. Still, there is a clear limit to how many brightness levels people can reliably distinguish since there must be quite a difference in brightness between each step on the scale to avoid confusion. So

again, more than 4-5 alternative equipment types allowed in a process model would cause problems. This could be mitigated, however, by redundant combination with text tags, as shown in row 6, which is anyway considered a positive thing according to Moody's principle of *dual coding* [29], and here it would be especially important to users with impaired colour vision.

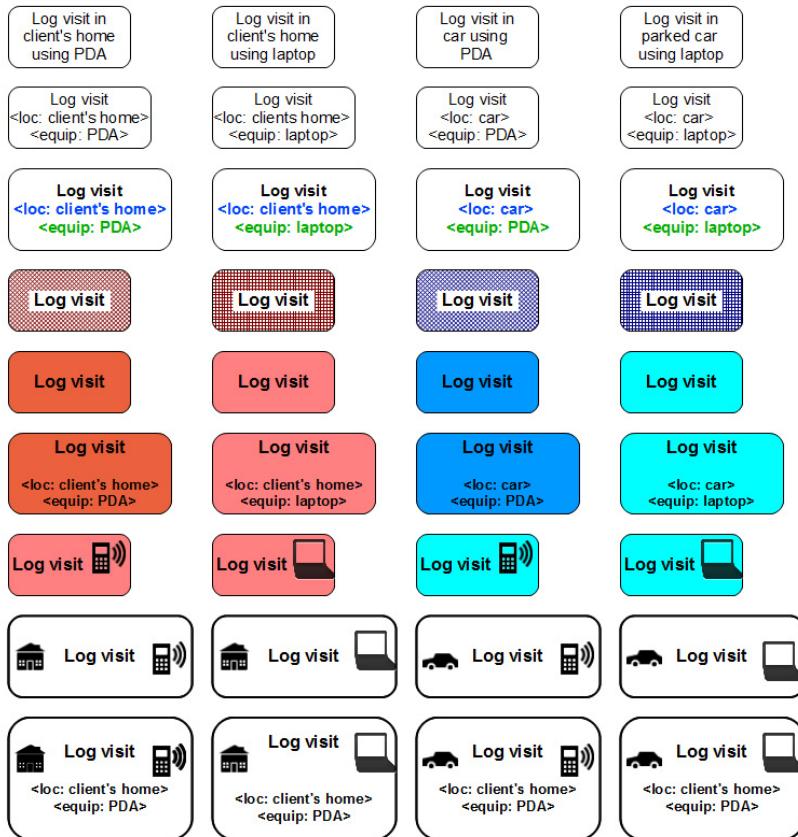


Fig. 3. Alternatives only making changes internally in the activity nodes. In case of b/w printing, row 3 has blue <loc> tags and green <equip> tags, and in rows 4-7 the two leftmost columns are red and the two rightmost are blue.

Row 7 still uses colour for location in the same ways as the rows above, but now combines this orthogonally with the use of some icons for equipment. As also pointed out in earlier works such icons are possibly the best alternative when it comes to semantic transparency, as they have what Moody would call an intuitive relationship to the concepts to be captured, at least as long as intuitive icons exist for the concepts in question. Given the advantage of icons for semantic transparency, one could of course also imagine dropping colour, and instead show both location and equipment

by icons. Even if it might be quite intuitive in most cases which icons then relate to locations and which relate to equipment, this could sometimes be a source of confusion, especially for stakeholders who associate a particular type of equipment very much with a certain location or vice versa. The bottom rows in Figure 3 indicate always placing location icons in the left side of the node and equipment icons in the right. This should not collide with the current use of icons in the top left corner of the activity [32]. Thus, in this case we actually manage to utilize also planar variables to convey meaning, without any conflict with other usages of planar variables in BPMN. The final row again redundantly combines graphical means with textual tagging.

5 Evaluation Results

Table 1 shows an evaluation of the various alternatives shown in the Figure 3 of the previous section. The rows of the table show the various alternatives discussed in the previous section in relation to Figure 3, with the reference to the illustration indicated in the parenthesis (e.g., 3, 1 for text + text meaning Figure 3, row 1). The columns of the table reflect the principles by Moody discussed earlier (except the two principles we chose not to include in the evaluation), plus the extra criterion for minimal deviation from the original BPMN language. Hence the columns are SC = semiotic clarity, PD = perceptual discriminability, ST = semantic transparency, CM = complexity management, VE = visual expressiveness, DC = Dual Coding, GE = graphical economy, CF = cognitive fit, and finally MD = minimal deviation from mainstream notation).

A five point scale is used for grading, namely --, -, blank, +, and ++, indicating the range from a strong disadvantage, through neutral, to a strong advantage for the criterion in question. To the right we have indicated two sum columns; the first one called "Sum" assumes that all 8 criteria are equally important, while the Weighted Sum (WS) column gives double weight to the three leftmost criteria (semiotic clarity, perceptual discriminability and semantic transparency). In the following, the scores in Table 1 are explained criterion by criterion.

Table 1. Evaluation of alternatives, assuming small adaptation of BPMN

Location + equipment	SC	PD	ST	CM	VE	DC	GE	MD	Sum	WS
Text + text (3,1)	--	--				-	++	++	-1	-5
Tag + tag (3,2)	-	+	+	+		-	+	+	2	2
Tag + tag, w/col (3,3)			+	+	+		+	+	5	6
Colour + pattern (3,4)	++	+		++	++	-		-	5	8
Col. + brightness (3,5)	++	+		++	++	-		-	5	8
Col + bright + tags (3,6)	++	+		++	++	+		--	6	9
Colour + in-icon (3,7)	++	++	+	+	+	-		-	5	10
In-icon + in-icon (3,8)	++	++	++	+		-		--	5	11
In-ic+in-ic + tags (3,9)	++	++	++	+		+		--	6	12

Semiotic clarity (SC): Double minus to alternatives which offer no special representation for usage context at all. Single minus to alternatives which offers one symbol for context, but does not distinguish between location and equipment. Blank to alternatives which do distinguish between location and equipment, but only through text. Single plus to alternatives which distinguish between location and equipment graphically, but do not distinguish graphically between different locations and different equipment types (or at least not both of these). Finally double plus to alternatives showing graphical differences both between different locations and different equipment type. *Perceptual discriminability (PD):* Double minus to alternatives where there is no specific indication in the diagram that there is context information. Single minus to alternatives where it is indicated that there is context information, but only subtly so that it is hard to spot from a quick look at a diagram. Plus to alternatives who clearly distinguish visually between several different locations and equipment types. To get a double plus the approach must work for a fairly large number of different locations and equipments. *Semantic transparency (ST):* Here, the only ones getting a double plus are the alternatives with icon + icon, since icons intuitively reflect the concepts captured. A single plus therefore to approaches using one icon in combination with something else. Also a single plus to alternatives using tags. The remaining alternatives get a blank for having only an abstract (opaque) relationship to the concepts captured. No minuses given here, since alternatives that were really poor and counter-intuitive were discarded already before this tabular analysis (cf. Figure 2). *Complexity management (CM):* Double minus to alternatives that create a new node for each location and a new node for each equipment type. Single minus to those alternatives using one external node. Blank to those making no new nodes, but longer text labels that are more difficult to read. A single plus to approaches instead using tags or approaches stuffing internal icons into the nodes so that nodes get more crammed. Double plus to approaches which neither introduce neither more text nor more material inside the nodes. *Visual expressiveness (VE):* Double plus to approaches using two new visual variables which are not used in BPMN already. Single plus to those using one of these variables. Blank for usage of icons, although this is a kind of shape, the icons are different from other shapes used in BPMN. Minus for usage of a new shape, and double minus for introducing two new shapes. *Dual coding (DC):* Plus to all approaches combining text and graphical elements, minus to all which do not. *Graphical economy (GE):* Double minus to approaches that introduce a big number of free-standing new symbols. Single minus to those introducing more than one free-standing new symbol. Blank to approaches introducing only one new symbol, e.g. dedicated context shape, or to those introducing several new symbols, but then in a family, subordinate to an existing symbol (e.g., modifying existing process nodes by the introduction of in-icons or fills). Plus to those not introducing new graphical symbols, only new text tags, and finally double plus to those introducing no new elements in the language whatsoever. *Minimal deviation (MD):* Double plus to alternatives making no modification to the language whatsoever. Single plus to those only introducing text tags. Blank to those only introducing one new thing to learn. Single minus to those introducing two new things to learn, and double minus to those introducing three or more new things to learn.

6 Discussion and Conclusion

There are a number of threats to the validity of an analytical evaluation as presented in this paper. Although we have based our evaluation on a well-known framework published by another researcher and taken effort to score all approaches fairly, it will be impossible to totally avoid subjective opinion by the researchers, both in how to score each alternative for each criterion, how to weight criteria, and which notation alternatives to include and exclude in the first place.

As for the weighting of the evaluation criteria, the "Sum" column used equal weight (1) for all criteria. Since Moody does not suggest any weighting, this might seem a natural choice. However, it can also be observed that the criteria are not completely orthogonal, and it could be possible to argue that especially the three first criteria are more fundamental than the other ones, in reflecting essential properties that a visual language should have. Some of the other principles could be said to be more secondary factors for achieving the same objectives. Finally, while semiotic clarity, perceptual discriminability and semantic transparency are more fundamental aspects of the visual language, some other criteria would be more up to the tool support than just the language itself, especially complexity management. Hence, we also introduced the WS column where the three criteria appear more fundamental. On the other hand, if the approach should be implemented, some tools cannot allow modifying icons although many modern tool-environments support such extensions, updating the meta-models supported.

Several candidate notations appear as approximately equal for the "Sum" column, the alternatives colour + brightness + tags and in-icon + in-icon + tags both having 6 points, and several other alternatives following closely on 5 points. For the weighted sum (WS column), the internal icon alternative more clearly emerges as the best according to the analytical evaluation, gaining on the alternatives using colour mainly because of its better semantic transparency. Another advantage of the icon approach compared to approaches using colour, brightness or similar is in cases where there is a need to show several locations or equipment types for the same process node, e.g., several alternative equipment types are allowed based on user preference, or even cases where two devices are needed at the same time to accomplish a task.

Hence, the top score for the internal icon approach makes sense also intuitively, as it scales better with several alternative usage contexts, has good semantic transparency, and still avoids the introduction of extra nodes and links which makes the diagram structurally more complex. Still, further empirical evaluations will be necessary to investigate whether the notations that did the best in this analytical evaluation will really be better than other alternatives in practice. Such evaluations could include both experiments similar to those already done in [4,5] and case studies in more industrial settings. In experiments the cross-cultural differences and variations of information metabolism (in terms of socionics) can also be taken into account.

References

1. Fowler, M., Scott, K.: UML Distilled: A brief guide to the standard object modeling language, 2nd edn. Addison-Wesley Professional, Reading (2000)
2. Jacobson, I., et al.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, Boston (1992)
3. Gopalakrishnan, S., Sindre, G.: Alternative Process Notations for Mobile Information Systems. In: Popplewell, K., et al. (eds.) Enterprise Interoperability IV. Making the Internet of the Future for the Future of Enterprise, pp. 13–23. Springer, London (2010)
4. Gopalakrishnan, S., Krogstie, J., Sindre, G.: Adapting UML Activity Diagrams for Mobile Work Process Modelling: Experimental Comparison of Two Notation Alternatives. In: van Bommel, P., Hoppenbrouwers, S., Overbeek, S., Proper, E., Barjis, J., et al. (eds.) PoEM 2010. LNBP, vol. 68, pp. 145–161. Springer, Heidelberg (2010)
5. Gopalakrishnan, S., Krogstie, J., Sindre, G.: Adapted UML Activity Diagrams for Mobile Work Processes: Experimental comparison of colour and pattern fills. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBP, vol. 81, pp. 314–331. Springer, Heidelberg (2011)
6. Gopalakrishnan, S., Sindre, G.: Diagram Notations for Mobile Work Processes. In: Johannesson, P., Krogstie, J., Opdahl, A.L. (eds.) PoEM 2011. LNBP, vol. 92, pp. 52–66. Springer, Heidelberg (2011)
7. Gopalakrishnan, S., Krogstie, J., Sindre, G.: Extending Use and Misuse Cases to Capture Mobile Information Systems. In: Fallmyr, T. (ed.) Norsk Konferanse for Organisasjoner Bruk av Informasjonsteknologi (NOKOBIT). Tapir, Trondheim (2011)
8. Gopalakrishnan, S., Krogstie, J., Sindre, G.: Extending Use and Misuse Case Diagrams to Capture Multi-Channel Information Systems. In: Abd Manaf, A., Zeki, A., Zamani, M., Chuprat, S., El-Qawasmeh, E. (eds.) ICIEIS 2011, Part I. CCIS, vol. 251, pp. 355–369. Springer, Heidelberg (2011)
9. Krogstie, J., Lyytinen, K., Opdahl, A.L., Pernici, B., Siau, K., Smolander, K.: Mobile Information Systems - Research Challenges on the conceptual and logical levels. In: Olivé, Å., Yoshikawa, M., Yu, E.S.K. (eds.) ER 2003. LNCS, vol. 2784, pp. 124–135. Springer, Heidelberg (2003)
10. Kristoffersen, S., Ljungberg, F.: Mobile Use of IT. In: 22nd Information Systems Research Seminar in Scandinavia, Jyväskylä, Finland (1999)
11. Esbjörnsson, M.: Work in Motion: Interpretation of Defects along the Roads. In: The 24th Information Systems Research Seminar in Scandinavia, Bergen, Norway (2001)
12. Veijalainen, J.: Developing Mobile Ontologies; who, why, where, and how? In: International Conference on Mobile Data Management. IEEE, Manheim (2007)
13. Dourish, P.: Re-space-ing place: “place” and “space” ten years on. In: Proceedings of the 2006 20th Anniversary Conference on Computer supported Cooperative Work, pp. 299–308. ACM, Banff (2006)
14. Harrison, S., Dourish, P.: Re-place-ing space: the roles of place and space in collaborative systems. In: Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work, pp. 67–76. ACM, Boston (1996)
15. Nossom, A., Krogstie, J.: Integrated Quality of Models and Quality of Maps. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009. LNBP, vol. 29, pp. 264–276. Springer, Heidelberg (2009)
16. Decker, M., et al.: Modeling Mobile Workflows with BPMN. In: International Conference on Mobile Business/Global Mobility Roundtable (2010)

17. Decker, M.: Modelling Location-Aware Access Control Constraints for Mobile Workflows with UML Activity Diagrams. In: 3rd International Conference on Mobile Ubiquitous Computing, Systems and Technologies, Siema, Malta (2009)
18. Baumeister, H., Koch, N., Kosciuczenko, P., Wirsing, M.: Extending Activity Diagrams to Model Mobile Systems. In: Akşit, M., Mezini, M., Unland, R. (eds.) NODe 2002. LNCS, vol. 2591, pp. 278–293. Springer, Heidelberg (2003)
19. Kosciuczenko, P.: Sequence Diagrams for Mobility. In: Olivé, À., Yoshikawa, M., Yu, E.S.K. (eds.) ER 2003. LNCS, vol. 2784, pp. 147–155. Springer, Heidelberg (2003)
20. Kusek, M., Jezic, G.: Extending UML Sequence Diagrams to Model Agent Mobility. In: Padgham, L., Zambonelli, F. (eds.) AOSE 2006. LNCS, vol. 4405, pp. 51–63. Springer, Heidelberg (2007)
21. Mendling, J., Recker, J., Reijers, H.A.: On the Usage of Labels and Icons in Business Process Models. *International Journal of Information System Modeling and Design* 1(2), 40–58 (2010)
22. Wand, Y., Weber, R.A.: On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. *Journal of Information Systems* (1993)
23. Recker, J., et al.: Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. In: 16th Australasian Conference on Information Systems, Sydney (2005)
24. Lindland, O.I., Sindre, G., Sølvberg, A.: Understanding Quality In Conceptual Modelling. *IEEE Software* 11(2), 42–49 (1994)
25. Krogstie, J.: Model-based development and Evolution of Information Systems. Springer (2012)
26. Aagesen, G., Krogstie, J.: Analysis and design of business processes using BPMN. In: *Handbook on Business Process Management*. Springer (2010)
27. Recker, J., Rosemann, M., Krogstie, J.: Ontology-versus pattern-based evaluation of process modeling languages: a comparison. *Comm. Assoc. Inform. Syst.* 20, 774–799 (2007)
28. Wahl, T., Sindre, G.: An Analytical Evaluation of BPMN Using a Semiotic Quality Framework. In: Siau, K. (ed.) *Advanced Topics in Database Research*, vol. 5, pp. 94–105. IGI Global (2006)
29. Moody, D.L.: The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering* 35, 756–779 (2009)
30. Bertin, J.: *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, Madison (1983)
31. Genon, N., Heymans, P., Amyot, D.: Analysing the cognitive effectiveness of the BPMN 2.0 visual notation. In: Malloy, B., Staab, S., van den Brand, M. (eds.) SLE 2010. LNCS, vol. 6563, pp. 377–396. Springer, Heidelberg (2011)
32. Silver, B.: *BPMN Method & Style*, 2nd edn. Cody-Cassidy Press (2012)

Capturing Decision Making Strategies in Enterprise Architecture – A Viewpoint

Georgios Plataniotis^{1,2,3}, Sybren de Kinderen^{1,3}, and Henderik A. Proper^{1,2,3}

¹ Public Research Centre Henri Tudor, Luxembourg, Luxembourg

² Radboud University Nijmegen, Nijmegen, The Netherlands

³ EE-Team, Luxembourg, Luxembourg*

{georgios.plataniotis, sybren.dekinderen, erik.proper}@tudor.lu

Abstract. Enterprise Architecture modeling languages describe an enterprise holistically, showing its business products and services and how these are realized by IT infrastructure and applications. However, these modeling languages lack the capability to capture the design rationale for decisions that lead to specific architectural designs. In our previous work we presented the EA Anamnesis approach for capturing architectural decision details. In this paper, we extend the EA Anamnesis approach with a viewpoint that captures and rationalizes decision making strategies in enterprise architecture. Such a viewpoint is useful because it helps enterprise architects reconstruct the decision making process leading up to a decision and understand how and under which circumstances this decision was made. For example, under time pressure an architect may rely on heuristics instead of examining the decision problem in depth. More specifically, we contribute: (1) a metamodel for capturing decision making strategies, which is grounded in established decision making literature, (2) an illustrative example showcasing the potential usefulness of capturing the decision making process.

Keywords: Enterprise Architecture, Decision making strategies, Design Rationale, Decision Capturing.

1 Introduction

Enterprise Architecture (EA) [1, 2] is a steering instrument that connects an organization's IT infrastructure and applications, to the business processes they support and the products/services that are in turn realized by the business processes. Such a holistic perspective on an enterprise helps clarify the business advantages of IT, analyze cost structures and more [3].

A variety of Domain Specific Languages for the modeling of Enterprise Architectures have been created, such as the Open Group standard ArchiMate [4].

* The Enterprise Engineering Team (EE-Team) is a collaboration between Public Research Centre Henri Tudor, Radboud University Nijmegen and HAN University of Applied Sciences (www.ee-team.eu).

While Enterprise Architecture modeling languages allow for modeling an enterprise holistically, the design decisions behind the resulting models are often left implicit. Although we should be careful with the analogy, experience from the field of software architecture shows that leaving design rationales implicit leads to ‘Architectural Knowledge vaporization’ (cf. [5]). This means that, without design rationale, design criteria and reasons that lead to a specific design are not clear. Also, alternatives that were considered during the design process are not captured.

Among others, such lack of transparency regarding design decisions can cause design integrity issues when architects want to maintain or change the current design [6]. This means that due to a lacking insight of the rationale, new designs are constructed in an adhoc manner, without taking into consideration constraints implied by past design decisions. Also, according to a survey for software architecture design rationale [7], a large majority of architects (85,1%) admitted the importance of design rationalization in order to justify designs. Another interesting finding of this survey was that architects declared that after some time they frequently forget their own decisions. Moreover, anecdotal evidence from six exploratory interviews we conducted with senior enterprise architects, suggests that Architects are often external consultants. This situation increases the architectural knowledge gap of the Enterprise Architecture. The successor architect tries to understand and analyze the architecture by searching through architectural designs and unstructured information of requirements documentation.

In our previous work [8] we introduced an approach for the rationalization of enterprise architectures by capturing EA design decision details. We refer to this approach as EA Anamnesis, from the ancient greek work *ανάμνησις* (/ænæm'ni:sis/), which denotes memory and repair of forgetfulness. The metamodel is grounded in Decision Representation Language (DRL) [9] and is based on similar approaches from the software engineering domain. At this stage, EA Anamnesis complements the ArchiMate modeling language [4] by conceptualizing decision details (alternatives, criteria, impacts) and by grouping EA decisions in three different enterprise architecture layers (Business, Application, Technology) in accordance with the ArchiMate specification.

However, decision details captured by EA Anamnesis do not describe explicitly the *decision making process* during the architectural design. Although our approach captures alternatives and criteria of EA decisions, it does not provide information on *how* the decision process was executed. EA Anamnesis does not capture what decision strategy was used, and what factors led to the adoption of such a strategy. Yet, if considerations during the decision process are not captured, one loses the insight of the factors that also contributed towards taking the actual decision. For example, a decision could have been made under time pressure, and as such, a heuristic decision strategy may have been used instead of considering all criteria and their respective importance.

In this paper we extend the EA Anamnesis approach in order to capture the decision making process. Specifically, we contribute: (1) a decision making

strategy viewpoint metamodel that captures the basic characteristics of decision making process (decision strategies, criteria). The concepts from this metamodel are grounded in established decision making literature [10–14]. (2) we illustrate with a fictitious scenario how such a viewpoint can be used to capture the decision making process.

Our approach helps an enterprise architect (probably not the actual designer) to reconstruct the decision making process and understand how his predecessor made an EA decision. It does so by making explicit, amongst others, the decision making criteria, the respective importance of criteria, the used decision making strategy and the rationale for selecting a decision making strategy. This would nicely complement our EA Anamnesis approach in the sense that it would allow for comparing the results of a decision with the criteria and used decision making process leading up to the decision. In such a way, architects can compare past decision making criteria to observed outcomes to learn from captured decisions.

While we acknowledge that decision making often involves multiple parties [1], this paper focuses on a single decision maker: the enterprise architect.

This paper is structured as follows. Section 2 presents the background literature in decision strategies and challenges, Section 3 introduces the decision strategy metamodel, while Section 4 illustrates the use of our approach with an insurance industry example. Finally Section 5 concludes.

2 Background Literature

The work reported in this paper is based on established literature of decision making strategies. This section reports on the two streams that were examined: (1) actual decision making strategies, and (2) factors that influence the decision making.

2.1 Decision Making Strategies

Decision making strategies generally fall in two main categories: compensatory and noncompensatory [10–12, 15]. We briefly explain these strategies with a car buying example. In this example, a customer selects a car based upon the criteria ‘color of car’, ‘carbon emission’, ‘small size of car’ and ‘gasoline consumption’.

In compensatory decision making [10], alternatives are evaluated exhaustively, taking *all* criteria and their trade-offs into consideration. Criteria with high values compensate for criteria with lower values. Finally, the alternative with the highest score is selected. For our car buying example, this implies that a customer considers all four criteria ‘color of car’, ‘carbon emission’, ‘small size of car’ and ‘gasoline consumption’. For example: s/he can state that ‘color of car’ is of high importance, and ‘carbon emission’, ‘small size of car’ and ‘gasoline consumption’ are of less. By doing so the customer then selects a car that best complies with all these criteria.

Compensatory strategies aim to provide the best possible decision outcomes given the decision data at hand. However, compensatory strategies require full

information on how alternatives score on all criteria, and they are time consuming [10].

Noncompensatory strategies [10], on the other hand, are consistent with the concept of bounded rationality. This means that the rationale to make a decision is limited by factors such as hard constraints, time constraints and the cognitive load of the decision maker. As such, noncompensatory strategies evaluate alternatives heuristically, using only a limited number of criteria and trade-offs.

Considering a noncompensatory decision strategy for our car buying example, let us now assume that the customer lives in the city and selects between two cars: a small and a big one. Now, ‘small size of car’ is a hard constraint for the customer due to the limited parking space available in the city. Therefore, regardless of the criteria ‘carbon emission’, ‘color of car’, and ‘gasoline consumption’ s/he excludes the big car from her/his choice set.

The main characteristics of noncompensatory strategies are twofold: (1) they reduce the decision making effort, (2) they are not demanding regarding the information needed to make the decision. As such, it is a common practice for decision makers to use noncompensatory strategies in situations (time stress, hard constraints) the limitations of which affect the decision making process [16]. However, by definition decision makers do not take all criteria into account when using noncompensatory decision strategies.

Last but not least, in some cases the use of a combination of compensatory and noncompensatory decision strategies (a hybrid) is required [15, 17, 18]. For example, a decision maker starts his evaluation process by excluding alternatives that do not meet certain noncompensatory criteria, and only thereafter evaluates the remaining alternatives with a compensatory strategy.

2.2 Factors Influencing the Decision Making

Decision making, in particular the choice for a decision making strategy, is influenced by factors such as time, information completeness, cognitive load of the decision maker, and more [13].

We describe briefly those factors and how they influence the decision making process.

Time stress: One of the most common situations in decision making processes is time stress [14]. Decision makers under time pressure must take critical decisions. Usually these decisions are made last moment, in an adhoc manner, and without sufficient rationalization.

Ill-structured problems: A decision problem is often complex in terms of cause-effect relations, correlations and feedback loop between relevant factors [14]. As such, decision problems are difficult to understand, also in terms of the impacts and outcomes that they have.

Information incompleteness: meaning that in practice information can be ambiguous or even missing [14].

Shifting, ill-defined, or competing goals: A decision maker can have conflicting goals during the decision making process. Decision maker should weight appropriately each of these goals in making a decision [20].

Action and feedback loops: Decision making contains a series of loops that the decision maker should deal with [14]. Early mistakes and poor information generate decisions that should be reexamined. For example violations in architectural design are not disclosed early enough and this implies that the decision making process should be repeated.

High stakes: Decision makers have also to cope with high risk decisions, especially when the problem they are called to solve is of high importance [14].

Multiple player situations: When multiple stakeholders are involved in a decision making process the situation gets more complicated [14]. Stakeholders have different interests, goals and expectations from a specific decision. Another common issue is the lack of shared understanding between stakeholders for a particular problem. Multiplayer situations may result in delays in the decision making possibly leading to a revision of the decision, with high cost impact [21].

Organizational goals and norms: Organizations operate under specific goals and norms [14]. Decision makers should make decisions in the context of these goals and norms and should avoid making decision based only on their personal preferences. Decision makers, regardless of their personal preferences, must evaluate alternatives with criteria that the organization sets.

Note that our approach does not provide rationalization support for decision strategies with multiple decision makers. As we mentioned before, this paper focuses on single decision maker environments.

3 The Decision Strategy Viewpoint

In this section we introduce the motivation for the decision making strategy viewpoint. In line with [22], we use a separate viewpoint for capturing the decision strategy to concentrate on concerns that are of interest for the decision making process itself (such as time pressure, the use of heuristics in decision making). Thereafter, we discuss the conceptualization of the decision making strategies and how these concepts extend EA Anamnesis approach for EA rationalization.

3.1 Motivation

Decision making for architects can be challenging. This is due to, amongst others, the cross organizational nature of enterprise architecture: inherently, architects involve stakeholders from different backgrounds and with differing concerns [1].

As a result, Enterprise architects as decision makers have to cope with various challenges in the decision making process such as time stress, ill structured problems, uncertain dynamic environments and others [14]. As we mentioned in Section 2.2, these factors affect the decision making process. Decision makers should be able to adopt a decision strategy that is appropriate for these challenges.

The proposed viewpoint captures and reconstructs decision making processes for EA decisions. By comparing the captured decision strategy with the observed outcome of a decision, architects can understand if the decision making strategy

was successful or had negative consequences. This helps enterprise architects to follow or avoid decision making practices. For example: the viewpoint presented in this paper can be used to capture whether a decision was made under time pressure. Subsequently, if the outcomes of a decision was negative, we can use this information to make transparent *why* the decision was made.

3.2 Decision Strategy Viewpoint Metamodel

Figure 1 presents the metamodel of the Decision Strategy viewpoint. The idea of using different viewpoints for representing different types of information was taken from the ISO/IEC/IEEE 42010 standard for Architectural descriptions in Systems and Software Engineering [23]. The decision strategy viewpoint focuses on capturing (1) decision making strategies that were used during the architectural design process for a specific EA decision, (2) the rationale behind this specific decision strategy choice, and (3) available alternatives and criteria that were taken into account.

Decision-Making Strategy: This is the central concept of our viewpoint. It captures the decision making strategy used by the enterprise architect to (1) evaluate the alternatives, and make the actual EA decision. As we mentioned in Section 2.1, decision strategies are characterized as compensatory, noncompensatory, or as a hybrid of these two. In our metamodel, we specify this as follows:

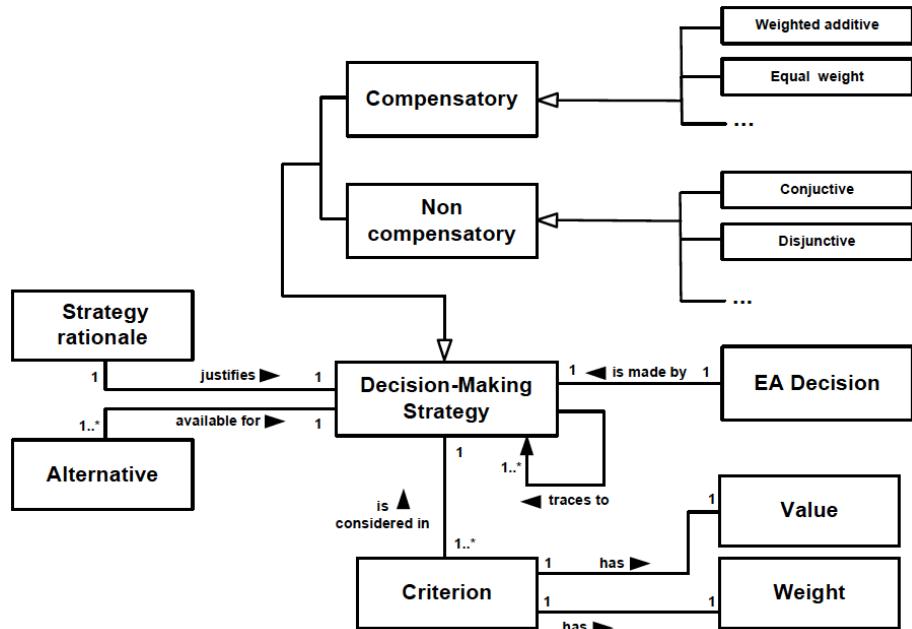


Fig. 1. Metamodel of Decision Making Strategy viewpoint

– **Compensatory strategy**

- Weighted additive (WADD): In WADD strategies the criteria that evaluate the alternatives have different weights. The score of each alternative is computed by multiplying each criterion by its weight and then by taking the sum of these values. The alternative with the highest score is chosen by the decision maker [15].
- Equal weight: The score of each alternative is calculated by the same way as WADD strategies. The difference is that criteria have the same weight [15].

– **Noncompensatory strategy**

- Conjunctive: In conjunctive strategies, alternatives that fail to comply with a minimum threshold level of one or more criteria, are immediately excluded from the decision maker's choice set [15].
- Disjunctive: In this strategy alternatives are evaluated based on the maximum threshold level of one or more criteria. Those which fail to meet the maximum level, are excluded from the choice set [15].

In line with Section 2.1 a hybrid decision strategy is supported by our metamodel. The relationship ‘trace to’ signifies the combination of two or more decision strategies during the decision making process.

We should also mention that there is no restriction in the use of additional decision strategies. We include a set of common decision strategies, but we also denote in the strategy viewpoint metamodel that more decision strategies can be supported.

Strategy Rationale: In a decision making process, the architect not only has to choose amongst some alternatives (actual decision making process), but has also to select the decision strategy that satisfies his current evaluation needs. Actually, this concept represents the rationale for the decision strategy that was selected for the evaluation process. This is what is referred as metadecision making, decision making about the decision process itself [24].

As we discussed in Section 2.2, different factors affect the decision making process and decision makers should adjust accordingly their decision making strategy. The concept of a strategy rationale enables a decision maker to justify the reasons for his metadecision. For example, budget issues may be a strategy rationale for selecting a noncompensatory decision making strategy.

Criterion: Criteria play an important role in the decision strategy viewpoint. Depending on the decision strategy that was used for the evaluation process, criteria can be compensatory or noncompensatory. For example, if a disjunctive strategy was used, the criteria that were used for the evaluation with this strategy are disjunctive. Furthermore, the concepts **value** and **weight** of criterion are included in our viewpoint. Value concept represents the value that the decision maker assigns to this criterion during the evaluation process and weight concept represents the importance of this criterion. Weight concept is used in WADD strategies. This gives the ability to the architect to trace back the decision making process and analyze the value as well as the importance of each criterion

during the evaluation process. By capturing the type, value and weight of criteria, stakeholders that analyze in depth the architecture, can understand which criteria had a determinant role in the selection process and on which strategy they were based.

EA Decision: An EA decision represents the actual decision that was made after the evaluation process. EA decision is the central concept for EA Anamnesis approach [8]. As such, EA decision can act as a bridging concept between our EA Anamnesis approach (see Introduction) and the Decision Strategy viewpoint. The EA decision captures the actual decision made, while the decision strategy viewpoint describes the strategy leading up to the decision.

Alternative: The concept of alternative represents the available choices that are under evaluation by using a specific decision making strategy.

4 Illustrative Example

We now show how the EA Anamnesis approach can be used to capture architectural decision details as well as decision making strategy details by using the proposed strategy viewpoint. The strategy viewpoint enhances the rationalization information that EA Anamnesis provides. For illustration purposes, we use an insurance company case study presented in our previous paper [8]. We aim to illustrate that the proposed strategy viewpoint can assist Enterprise Architects in understanding not only architectural details of each EA decision, but also how decision making process for this specific EA decision was done.

4.1 ArchiInsurance: Moving to an Intermediary Sales Model

ArchiInsurance is an insurance company that sells insurance products using a direct-to-customer sales model. The company used this disintermediation scheme to reduce its operations and product costs.

Although, disintermediation reduces operational costs, the use of intermediaries in insurance sector is very important because they provide accurate risk customer profiles [25]. ArchiInsurance management decides to adopt this practice and to change its selling model to intermediary sales. The role of the Insurance broker is added to the business operation of the company.

4.2 Capturing a Decision Process for ArchiInsurance

In our scenario, an external architect called *John* is hired by ArchiInsurance to change the Enterprise Architecture and analyze the impacts that the intermediary sales of insurance has on the company.

John uses the EA modeling language ArchiMate to capture the impacts that selling insurance via an intermediary has in terms of business processes, IT infrastructure and more.

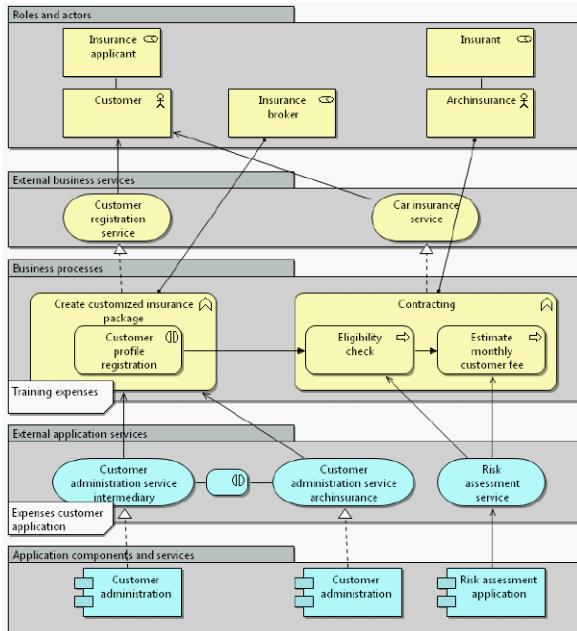


Fig. 2. ArchiSurance intermediary EA model

The resulting ArchiMate model is depicted in Figure 2. The initial ArchiMate model (before the transformation) is left out because of space limitations. Please refer to our previous work [8] for this EA model.

Here we see for example how a (new) business process ‘customer profile registration’, owned by the insurance broker (ownership being indicated by a line between the broker and the business process), is supported by the IT applications ‘customer administration service intermediary’ and ‘customer administration service ArchiSurance’.

However, John (by using ArchiMate) can not capture the rationale behind this model. For example, he captures the change for the different application architecture that supports the new business process, but he is not able to capture the justification for his decision. To capture design rationales behind the ArchiMate model, John relies on the EA Anamnesis approach (our previous work, see [8]).

For this simplified scenario 13 Architectural decisions were made. Table 1 shows an example application of the EA Anamnesis approach (EA decision 13). As it can be observed, decision facets such as the decision rationale (why the decision was taken), criteria (factors, such as cost), observed impact (ex-post) are captured. For further details, see [8]. However, what we lack in the EA Anamnesis approach is the idea of the decision making strategy leading up to the captured decision. For example: we cannot see if a decision was taken under time pressure, or what decision strategy was used to make a decision based upon the importance and tradeoffs of available criteria (as discussed in Section 2).

Table 1. EA decision 13 details

Title:	Acquisition of COTS application B
EA issue:	Current version of customer administration application is not capable to support maintenance and customers administration of intermediaries application service
Decision Maker:	John
Layer:	Application
Intra-Layer dependent Decisions:	EA decision 10
Inter-Layer dependent Decisions:	None
Alternatives:	COTS application A COTS application C Upgrade existing application (inhouse)
Rationale:	Scalability: Application is ready to support new application services
Criteria:	Customized reports capability, interoperability, scalability, cost
Observed Impact:	Interoperability issues. Application COTS B can not communicate with existing applications of some insurance brokers.

Therefore, we now replay the decision process leading up the the creation of the EA decision captured in Table 1. In doing so, we illustrate the EA Decision making strategy viewpoint.

Capturing a Decision Making Process: For the purposes of this paper, we focus on capturing and analyzing decision making strategies. Therefore, we assume that John is a single decision maker, who is capable to identify the above concepts and he has full information to evaluate them. We thus abstract away from identifying the specific alternatives, criteria and their respective scores (as briefly touched upon in Section 2.2).

To start the decision making process, John based on the requirements of the new business process, defines the criteria that the new application should satisfy (the criteria for application selection are grounded in [26]).

For our illustrative example, John considers that the most important architectural criteria are ‘customized reports capability’, ‘interoperability’ and ‘scalability’. Based on these criteria he identifies four alternatives to choose from: three alternative Commercial Off-The-Shelf (COTS) applications and one to upgrade the existing application in house.

Let us also assume that John receives a constraining budget reduction of 10000\$ for the acquisition of new IT systems.

John is now faced with a hybrid decision strategy: on the one hand, he wants to carefully evaluate the four alternatives on the criteria ‘customized reports capability’, ‘interoperability’ and ‘scalability’ (via a compensatory strategy), but on the other hand John has to account for the hard constraint of ‘budget limitation’ (via a non-compensatory strategy).

At this time John uses the decision viewpoint to capture and justify his strategy selection, as well as the alternatives and criteria of his decision problem. For the noncompensatory part, John wants to discard all alternatives that fail to meet the cost criterion. Because of this hard constraint, he chooses a disjunctive noncompensatory strategy (for an explanation, see Section 3.2) to exclude from his choice set alternatives that exceed the maximum value of one or more criteria.

Table 2 summarizes the score of each alternative. COTS application C is eliminated from the choice set because it failed to meet the maximum cost requirement.

As we mentioned before, disjunctive noncompensatory strategies evaluate alternatives using a maximum threshold level on one or more criteria. In this example the disjunctive criterion is ‘cost’. The alternatives ‘COTS A’, ‘COTS B’ and ‘Upgrade application’ comply with this criterion (Table 2) and will be evaluated further in the next step of the decision making process. ‘COTS C’ cost exceeds the maximum limit and is eliminated from the choice set. For noncompensatory strategies, alternatives either comply or not to some criteria and their score are Boolean data types. The scores of the alternatives are also captured based on our metamodel.

For the compensatory part, John evaluates the three remaining alternatives based on the values and the weight of each of the criteria.

Scalability is the most important factor because, according to John, this application should be able to support changes in the business processes of ArchiSurance. This is important in order to support the addition of extra intermediaries. Customized reports capability and interoperability are also important, but not as important as scalability.

Given the fact that criteria that evaluate alternatives have different weights, John selects the use of a weighted additive compensatory strategy. At this moment John captures again his decision strategy as well as the weights and the values of the compensatory criteria. The score of each alternative is calculated by multiplying the value of each criterion by its weight, and then by taking the

Table 2. EA decision 13 noncompensatory disjunctive strategy

Alternatives	cost	score
COTS A	9000\$	1
COTS B	8000\$	1
COTS C	12000\$	0
Upgrade app	5000\$	1

sum of these values. Here, the weights range from 1 - not important - to 10 - important. The alternative with the highest score is chosen by the decision maker.

Table 3 shows us: (1) the criteria. ‘Scalability’, the most important criterion for John, has a weight of 10, while ‘Custom reports’ and ‘interoperability’ have weights of 7 and 5 respectively, (2) the score on a particular criterion for each alternative. For example: the alternative ‘COTS B’ scores 9 on ‘scalability’, whereas ‘Upgrade app’ scores 4. (3) the total score of each alternative. For example: ‘COTS B’ receives the highest score and as such, is selected by John.

Reflecting upon the Captured Decision Making Process. So far, we have illustrated the decision process as captured by John. Let us now illustrate how decision making process for EA decision 13 can be useful by the new enterprise architect, Bob.

Bob’s predecessor, John, captured the decision making process with the EA Anamnesis decision strategy viewpoint. Bob can now analyze (1) the used strategy, (2) why this strategy was selected, and (3) the importance of criteria for this evaluation process.

Figure 3 shows the decision making process for EA decision 13 based on the decision strategy metamodel. From the decision making process, Bob understands that a hybrid model was used. More specifically, he realizes that the criterion ‘cost reduction’ was used to discard an alternative, with the use of a disjunctive non-compensatory strategy. Furthermore, Bob observes that a compensatory weighted additive strategy was used to evaluate the remaining alternatives. He realizes that his predecessor used this strategy, because the criteria ‘customized reports capability’, ‘interoperability’ and ‘scalability’ did not have the same importance for the selection of an appropriate application that would support the new business process of ArchiSurance. He can also see the weight of each criterion, as well as the final score of each of the alternatives.

This reconstruction of the decision making process makes transparent how an EA design decision has been made. Amongst other, this transparency allows an architect to compare the outcome of an EA decision with the decision making criteria that led to this decision. As a result, s/he can learn which factors in the decision making process had a positive/negative impact to the EA design and follow/avoid these decision making practices for future decisions.

After a period of time, COTS application B does not have a sufficient performance due to interoperability issues. Bob, is asked by management to explain the choice for COTS application B. He can reconstruct and examine the decision

Table 3. EA decision 13 compensatory weighted additive strategy

Alternatives	custom reports	interoperability	scalability	score
COTS A	7x7	7x5	7x10	154
COTS B	8x7	3x5	9x10	161
Upgrade app	5x7	5x5	4x10	100

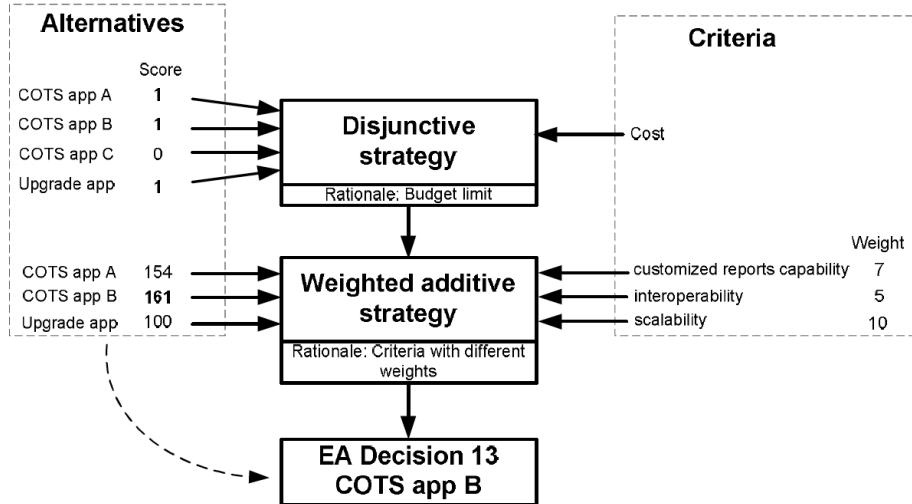


Fig. 3. Decision making process for EA decision 13

making process using Tables 2 and 3. First, he observes that COTS application C was eliminated because of budget issues. Second, from the weight assigned to the different criteria in Table 3, he observes that scalability was an important criterion for his predecessor to select COTS application B, but not interoperability. By examining the captured criteria and their weight as well as the observed impact of the EA Decision (Table 1) Bob can learn that interoperability is an important requirement for Archisurance enterprise architecture and should be weighted and compared against other criteria more carefully. For example: in a future decision making process, Bob can provide a weight of 7 or 8, instead of 5, to the criterion interoperability to better weight it against other criteria such as scalability.

5 Conclusion

In this paper we introduced a metamodel for capturing the decision making strategies in enterprise architecture. Furthermore, we argued why capturing the decision making strategy, next to the decision itself, is useful (1) by argumentation, (2) by means of an illustrative example.

For future research, first and foremost we intent to confront the illustrative examples of capturing and rationalizing a decision strategy to practitioners. As an example of such evaluation would be the estimation of the level of understanding of existing enterprise architectures and the decisions that led to them. Second, we aim at conducting case studies to capture architectural decision making strategies.

Furthermore, we aim to further extend our approach by facilitating the selection of decision making strategies during the decision making process. We deem such an extension relevant, because it helps the architect select an appropriate decision strategy based on the characteristics and constraints (such as time stress) imposed by the decision making environment.

Last but not least, one of our major challenges is to investigate the return of capturing effort for our approach. Our design rationale assists architects to better understand existing EA designs, but the effort of capturing this information might be a dissuasive factor. To address this issue our research will focus on ways to decrease the capturing effort. One way of doing this is by evaluating the actual practical usefulness of the concepts in the decision making strategy viewpoint. For example we capture the strategy rationale for selecting a decision making strategy, but it remains to be seen if the effort for capturing this, outweighs the received benefits.

Acknowledgments. This work has been partially sponsored by the *Fonds National de la Recherche Luxembourg* (www.fnr.lu), via the PEARL programme.

References

1. Op't Land, M., Proper, E., Waage, M., Cloo, J., Steghuis, C.: Enterprise architecture: creating value by informed governance. Springer (2008)
2. Hoogervorst, J.: Enterprise architecture: Enabling integration, agility and change. International Journal of Cooperative Information Systems 13(03), 213–233 (2004)
3. Lankhorst, M.: Enterprise architecture at work: Modelling, communication and analysis. Springer (2009)
4. The Open Group: ArchiMate 2.0 Specification. Van Haren Publishing (2012)
5. Jansen, A., Bösch, J.: Software architecture as a set of architectural design decisions. In: 5th Working IEEE/IFIP Conference on Software Architecture, WICSA 2005, pp. 109–120. IEEE (2005)
6. Tang, A., Jin, Y., Han, J.: A rationale-based architecture model for design traceability and reasoning. Journal of Systems and Software 80(6), 918–934 (2007)
7. Tang, A., Babar, M.A., Gorton, I., Han, J.: A survey of architecture design rationale. Journal of Systems and Software 79(12), 1792–1804 (2006)
8. Plataniotis, G., de Kinderen, S., Proper, H.A.: Ea anamnesis: towards an approach for enterprise architecture rationalization. In: Proceedings of the 2012 Workshop on Domain-Specific Modeling, DSM 2012, pp. 27–32. ACM, New York (2012)
9. Lee, J.: Extending the potts and bruns model for recording design rationale. In: Proceedings of the 13th International Conference on Software Engineering, pp. 114–125. IEEE (1991)
10. Einhorn, H.: The use of nonlinear, noncompensatory models in decision making. Psychological Bulletin 73(3), 221 (1970)
11. Payne, J.: Task complexity and contingent processing in decision making: An information search and protocol analysis. Organizational Behavior and Human Performance 16(2), 366–387 (1976)
12. Svenson, O.: Process descriptions of decision making. Organizational Behavior and Human Performance 23(1), 86–112 (1979)

13. Alenljung, B., Persson, A.: Portraying the practice of decision-making in requirements engineering: a case of large scale bespoke development. *Requirements Engineering* 13(4), 257–279 (2008)
14. Orasanu, J., Connolly, T.: The reinvention of decision making (1993)
15. Rothrock, L., Yin, J.: Integrating compensatory and noncompensatory decision-making strategies in dynamic task environments. *Decision Modeling and Behavior in Complex and Uncertain Environments*, 125–141 (2008)
16. Payne, J., Bettman, J., Johnson, E.: *The adaptive decision maker*. Cambridge University Press (1993)
17. Elrod, T., Johnson, R., White, J.: A new integrated model of noncompensatory and compensatory decision strategies. *Organizational Behavior and Human Decision Processes* 95(1), 1–19 (2004)
18. Jeffreys, I.: The use of compensatory and non-compensatory multi-criteria analysis for small-scale forestry. *Small-scale Forestry* 3(1), 99–117 (2004)
19. McAllister, D., Mitchell, T., Beach, L.: The contingency model for the selection of decision strategies: An empirical test of the effects of significance, accountability, and reversibility. *Organizational Behavior and Human Performance* 24(2), 228–244 (1979)
20. Ruhe, G.: Software engineering decision support: methodology and applications
21. Regnell, B., Paech, B., Aurum, A., Wohlin, C., Dutoit, A.: Requirements mean decisions!—research issues for understanding and supporting decision-making in requirements engineering. In: First Swedish Conference on Software Engineering Research and Practise: Proceedings. Citeseer (2001)
22. Nuseibeh, B., Kramer, J., Finkelstein, A.: A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering* 20(10), 760–773 (1994)
23. IEEE: Systems and software engineering – architecture description. ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), 1–46 (January 2011)
24. Mintzberg, H., Raisinghani, D., Theoret, A.: The structure of unstructured decision processes. *Administrative Science Quarterly*, 246–275 (1976)
25. Cummins, J., Doherty, N.: The economics of insurance intermediaries. *Journal of Risk and Insurance* 73(3), 359–396 (2006)
26. Jadhav, A., Sonar, R.: Evaluating and selecting software packages: A review. *Information and Software Technology* 51(3), 555–563 (2009)

Constructing Domain Knowledge through Cross Product Line Analysis

Ora Wulf-Hadash and Iris Reinhartz-Berger

Department of Information Systems,
University of Haifa, Haifa 31905, Israel
orawulf@gmail.com, iris@is.haifa.ac.il

Abstract. Nowadays many companies develop and maintain families of systems, termed *product lines* (PL), rather than individual systems. Furthermore, due to increase in market competition and the dynamic nature of companies' emergence, several PLs may exist under the same roof. These PLs may be independently developed taking into consideration different sets of products and requirements. Thus the developed artifacts potentially have a different and partial view of the domain. Moreover, future development and maintenance of the different PLs may require consolidating the various artifacts into a single coherent one. In this work, we present a method for constructing domain knowledge through cross PL analysis. This method uses similarity metrics, text clustering, and mining techniques in order to create domain models and recommend on improvements to the existing PLs artifacts. Preliminary results reveal that the method outcomes reflect human perception of the examined domain.

Keywords: Domain Analysis, Software Product Lines, Feature Diagrams, Feature Clustering, Feature Similarity, Feature Mining, Empirical Evaluation.

1 Introduction

Domain is considered “an area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area” [5]. Different systems or software products may be developed in the same domain. When developed in the same company, it may be effective and efficient to develop and maintain systems that share a common, managed set of features as families, called *product lines* (PL), rather than as individual systems. The field of *Software Product Line Engineering (SPLE)* [5] promotes the development and maintenance of artifacts that can be reused in families of related software-intensive systems. These artifacts, which are commonly termed *core assets*, specify and implement the common aspects of a PL and support and guide the specification and implementation of variable aspects. The core assets are further used for developing and assembling specific products in the PL, aiming to improve the quality and reduce development time and costs [5]. A common way to specify a core asset is a *feature model*, which captures the characteristics of a

given PL, as well as the relationships and constraints (dependencies) among these characteristics (features). Feature models are considered relatively simple, as they are technology-independent and include characteristics that are visible (and thus potentially understandable) to stakeholders, and especially to end-users.

The development of core assets in general and feature models in particular is a demanding task. It requires identification of common aspects, analysis of possible variations, and development of flexible, yet structured and well-defined, artifacts. These activities consume a lot of resources and involve many product- and production-related constraints [5]. Different studies have proposed methods for constructing feature models from textual descriptions [2], [26], source code [17], or existing products configuration [1], [20]. Other studies introduce manual domain analysis methods for utilizing past experience and knowledge in a certain domain (see, for example, [3], [8] for surveys of such methods), or offer support for automated or semi-automated domain analysis of UML-based artifacts, e.g., [12], [16], [19]. However, domain knowledge within these studies is constructed within the scope of a particular PL, having the potential of resulting with inaccurate and incomplete artifacts that represent partial views of the domain.

To enable the construction of wider and more accurate domain knowledge, we call in this paper for taking into consideration artifacts of different PLs in the same domain, observing three main sources for such artifacts. First, due to increase in market competition, companies cannot afford to focus on single PLs and need to develop several PLs for different customers, requirements, etc. Usually all these PLs are in a single domain, in which the company specializes, but include different common and variable aspects. As an example consider the domain of mobile phones. Samsung, for instance, develops several different PLs, including WAVE, Tocco, Shark, Genio, and Galaxy. The Galaxy family is further divided into Galaxy S, Galaxy Note, Galaxy ACE, and Galaxy Tab sub-families. Although these families are different from each other, many of their features are inherited from the domain of mobile phones or from the fact that they are all developed by Samsung. Second, many companies underwent mergers or acquisitions in the last two decades. These strategies can help an enterprise grow rapidly without “creating a subsidiary, other child entity or using a joint venture” [27]. A merger or an acquisition may yield different PLs that can be considered in the same domain. As an example in the mobile phones domain consider the merger of Sony and Ericsson in 2001 which yield the existence of different PLs originated from the two companies. Finally, different companies that develop PLs in the same domain may wish to cooperate for different reasons, willing to share their feature models that depict capabilities of their PLs rather than technologies and implementation aspects. A possible reason for such cooperation may be improving customer’s satisfaction by offering similar functional packages with the same interfaces.

To construct the domain knowledge, the method suggested in this paper gets a set of PLs artifacts, or more accurately feature models that represent different PLs in the same domain. The method further uses similarity metrics to compare features and their variations, adopts clustering techniques for grouping (and generalizing) similar features, and utilizes feature mining techniques for generating domain models and recommending improvements to the input PLs artifacts. The domain models provide a

holistic view on the domain of discourse and consolidate the knowledge extracted from the different PLs. The single coherent domain model can be used for a variety of purposes, including for suggesting improvements to the input PLs, for mapping similar features in various PLs, and for effectively managing the different PLs.

The rest of the paper is structured as follows. Section 2 reviews related work about domain analysis in the field of SPLE. Section 3 provides background about feature models. Section 4 describes and exemplifies the method, while Section 5 presents preliminary results regarding the method outputs. Finally, Section 6 concludes and refers to future research.

2 Related Work: Domain Analysis

Domain analysis is defined as “the process for capturing and representing information about applications in a domain, specifically common characteristics, variations, and reasons for variation” [5]. It supports collecting, organizing, and storing past experience and knowledge in building systems or parts of systems in a certain domain [6]. Different domain analysis methods have been suggested over the years [3], [8]. However, the manual nature of these methods made them time-consuming and error-prone.

Recently, several studies suggest (semi-)automated creation of core assets. These can be roughly divided into studies whose core assets are expressed as feature models and studies whose core assets are primarily expressed utilizing UML. The studies in the first group concentrate on creating feature models from textual descriptions, source code, or existing products configuration. Acher et al. [2], for example, present a semi-automated method for extracting the variability of a family of products from a set of product descriptions which represent valid combinations of features. Weston et al. [26] introduce an automated tool for creating feature models from requirements documents written in a natural language. The tool uses natural language processing techniques, as well as mining techniques for finding potential variable elements within the documents. Using source code, Rabkin et al. [17] present a static analysis method that extracts a list of configuration options for a program, utilizing standard points-to and call graph construction algorithms. Considering existing products configuration, She et al. [20] introduce a tool-supported approach for reverse engineering feature models aiming to significantly decrease the number of features that the modeler has to consider when creating feature models [26]. Acher et al. [1] present a tool-supported approach for reverse engineering architectural feature models. The reverse engineering process supports automatic extraction, aggregation, alignment, and projection of the input feature models. It is further capable of combining several sources of information, such as software architecture and plugin dependencies.

The studies in the second group focus on (semi-)automated generation of UML-based core assets. Nejati et al. [16], for example, offer matching statecharts statically through typographic matching, linguistic matching, and depth matching and behaviorally by checking bi-similarity between state-machines. Merging produces a combined model in which variant behaviors of the input models are parameterized using guards on their transitions. Giachetti et al. [12] introduce an automatic generation process of UML profiles that gets a domain-specific modeling language as an input, compares the meta-models to obtain the required UML extensions, and transforms the

integration meta-model into the corresponding UML profile. In [19], an approach for creating core assets from UML diagrams that specify individual software products is proposed. This approach defines linguistic, meta-informational, structural, and behavioral similarity metrics for performing matching, while integration is done by defining structured similarity groups and generalizing them into core assets.

All the above studies use the perspective of a single PL and thus the resultant model may partially and inaccurately represent the domain. Due to the reasons mentioned in the introduction, we aim to establish and enrich the constructed domain models by conducting cross PL analysis. This kind of analysis may also suggest improvements to the artifacts of the particular PLs, as well as enable interoperability among PLs of different companies (e.g., participating in the same consortium) and effective management of future development and maintenance. Before introducing the suggested method in details, we supply next the required background about feature models.

3 Feature Models

A *feature model* represents the common and variable aspects of products within a specific PL [14]. Feature models are usually composed of feature diagrams and descriptions. A *feature diagram* is a graphical notation for describing features and their relationships and dependencies. It is usually represented as a tree or a graph, where the nodes denote features and the edges – relationships and constraints (dependencies). *Feature descriptions* add information regarding the intention of the feature, its name, and its possible synonyms [24]. Feature descriptions may further refer to trade-offs, rationale, and justifications for feature selection [14]. Although feature descriptions add important information that is not shown in the diagrams, feature diagrams are commonly utilized alone, probably for simplicity.

Different formalisms have been suggested for feature diagrams. In the current work we use the definition presented in [2] which represent a feature diagram as a tree. Due to space limitations, we will not present the formal definition here, but will exemplify feature diagrams with a simple PL of mobile phones (see Figure 1). For future use, we will mark $f_1 \xrightarrow{\text{rel}} f_2$ to denote that f_2 is a (mandatory/optional/alternative/or) sub-feature of f_1 ; respectively $\text{rel} \in \{\text{mand.}, \text{opt.}, \text{xor.}, \text{or}\}$.

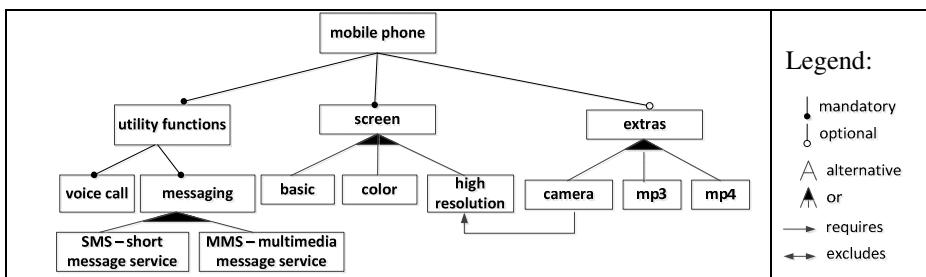


Fig. 1. An example of a feature diagram

4 Cross Product Line Analysis

4.1 Method Overview

The input to the method is a set of feature diagrams, $\{FD_i\}$. Each diagram is depicted as a tree of features and their relationships¹ and represents a different PL over the same domain². The input is processed in three main steps (see Figure 2). First, during the *Feature Similarity Analysis* step, the set of feature diagrams is analyzed using linguistic techniques for finding similar features that can serve as anchors for the next step of clustering. In the second step, *Feature Clustering*, an agglomerative clustering technique is used for creating groups (clusters) of similar features that may represent variants of the same features. Finally, in the third step, named *Cross PL Mining*, the method generates a domain model as well as recommendations for improvement in the input feature diagrams. In [30], we focus on the first two steps for carrying out commonality and variability analysis of different PLs. In the current paper we elaborate on the third step of generating domain models and recommending on appropriate improvements. However, for the sake of clarity, we briefly summarize in the sequel the two first steps (suggesting an improvement to the calculation of the feature similarity) and their intermediate outputs.

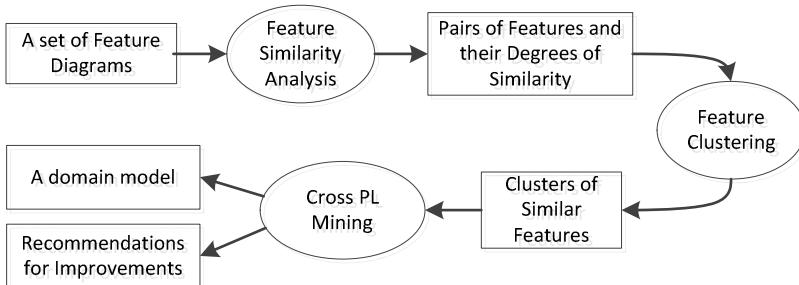


Fig. 2. An overview of the suggested method

4.2 Feature Similarity Analysis

Similarity between concepts and terms has been studied in the fields of ontology matching [4], [21] and data integration [23, 25]. In these fields, similarity measurements are primarily used for database schema integration, ontology merging, query answering, and data transformation. For these purposes, the suggested methods consider the semantic relatedness of concepts and terms, focusing on the common aspects of the different ontologies or data sources. In the current work, we adapt similarity

¹ At the current stage, the method ignores ('requires' and 'excludes') constraints.

² At the current stage, we do not validate whether the PLs are indeed over the same domain. If they are not, the method will still work, but its output will look like a union of the different PLs artifacts rather than a consolidated domain model.

measurements from these fields and further use them to analyze the variability among the different PLs (through the consequent steps of clustering and mining).

For measuring the similarity of features, the method uses their names and the context in which they appear (namely, their descendants and ascendants). In particular, for measuring similarity of the feature names, the method utilizes WordNet, which is a large, rich, freely available lexical database of English [28]. WordNet is general-purpose and hence can be used for different domains. The following definition is used for calculating feature name similarity.

Definition 1 (Feature Name Similarity). Let f_1 and f_2 be two features. *Feature name similarity*, $NSim$, is calculated as follows:

$$NSim(f_1, f_2) = \frac{\sum_{i=1}^m \max_{j=1..n} l_{t_i, u_j} + \sum_{j=1}^n \max_{i=1..m} l_{t_i, u_j}}{m + n}$$

Where: $t_1..t_m$; $u_1..u_n$ are the names of features f_1 ; f_2 , respectively (m and n are the numbers of words in the names of f_1 and f_2). l_{t_i, u_j} is Wu and Palmer's formula [29] for comparing two words. It is calculated as $\frac{2 * N_3}{N_1 + N_2 + 2 * N_3}$, where N_1 is the number of nodes on the shortest path from t_i to the least common super (LCS) concept of t_i and u_j in WordNet, N_2 is the number of nodes on the shortest path from u_j to LCS, and N_3 is the number of nodes on the shortest path from LCS to the root of WordNet.

For calculating the overall similarity of features, the method further uses, besides the feature name similarity, the immediate context of each feature, namely its sub-features and parent feature³. The input feature diagrams are scanned twice: once from the root to the leaves in order to calculate the (base) name similarity and increase the similarity of features whose descendants are similar and once from the leaves to the root in order to additionally increase the similarity of features whose descendants are similar. The following definition is used for calculating the overall feature similarity.

Definition 2 (Overall Feature Similarity). The overall feature similarity, or *feature similarity* for short, of features $f_1 \in F_1$ and $f_2 \in F_2$ takes into consideration the similarity in their names, the similarity of their sub-features ($CSim$), and the similarity of their parent features. It is calculated using the following formula:

$$\text{Sim}(f_1, f_2) = \frac{(m + 1) * CSim(f_1, f_2) + \max(CSim(f_1, f_2), \text{Sim}(f'_1, f'_2))}{m + 2}$$

Where f_1 , f_2 are sub-features of f'_1 , f'_2 , respectively; m is the number of f_1 sub-features multiplied by the number of f_2 sub-features; $CSim(f_1, f_2) = \frac{NSim(f_1, f_2) + \sum_{f''_1, f''_2} \max(NSim(f_1, f_2), CSim(f''_1, f''_2))}{m+1}$; f''_1 , f''_2 are sub-features of f_1 , f_2 , respectively.

Four important characteristics of the above formula (Definition 2) are: (1) the value of similarity is always in the range of 0 (completely different) to 1 (identical); (2) if the features have no similar sub-features neither similar feature parents, the overall

³ In [30], we refer only to name similarity and the similarity of sub-features. Here we additionally refer to the similarity of parent features, namely the similarity of features increases if they have similar parents.

similarity equals their name similarity; (3) the similarity of features increases proportionally to the degree of similarity of their ascendants; and (4) the similarity of features increases proportionally to the degree of similarity of their descendants.

As an example, consider the feature ‘messaging’ from Figure 1 and a feature named ‘sending utility’. The name similarity of the two features is 0.64. However, since ‘messaging’ has two children, SMS and MMS, and its parent is ‘mobile phone’, and ‘sending utility’ may have in another PL a sub-feature named ‘text message service’ and a parent also named ‘mobile phone’, the (overall) similarity of the two features may increase to 0.86. Note, however, that technological concepts, abbreviations, domain-specific acronyms may not be recognized by WordNet. To overcome this deficiency, we currently added the ability to import user-defined abbreviations and acronyms for certain domains (e.g., EMS, MMS, and SMS to respectively denote Enhanced, Multimedia, and Short Message Service). In the future we intend to improve this step and overcome the aforementioned limitations by using semantic similarity techniques, such as that based on Wikipedia⁴ [11].

4.3 Feature Clustering

The purpose of this step is to group “similar enough” features that may be termed and structured differently on the various input PLs. To this end, we use an agglomerative hierarchical clustering technique [15], which is a bottom-up approach that gets as a parameter the number of expected clusters and starts with putting each object (feature in our case) in a separate cluster. Then, the algorithm agglomerates (merges) in each iteration the closest pair of clusters by calculating the distance between different clusters. The algorithm continues until the number of expected clusters is reached. Starting with each feature in a different cluster, the algorithm prevents grouping features that are not similar enough. However, the algorithm requires determining the number of clusters a-priori. This number cannot be determined in our case as it varies depending on the size of the PLs and their degree of variability. Therefore, we modified the stopping criterion of the algorithm to merge two closest clusters as long as the distance between them is not larger than a pre-defined threshold. This way we ensure that too different features will not be put in the same cluster.

To determine the threshold for similar features, we use AdaBoost [9] which is a machine learning, adaptive algorithm. This algorithm is considered fast, simple, and easy to program [10]. In addition, it has no parameters to tune (except for the number of rounds) and it requires no prior knowledge about the weak learner and thus it can be flexibly combined with any method for finding weak hypotheses.

To calculate the distance between clusters, we use the complete-link distance type which measures the distance between the two farthest features of the clusters [13], [18]. This distance type is especially suitable to our case, since we aim to create a refined division of features to clusters and avoid merging features that are not similar enough to the same cluster. Note, however, that the method can be easily modified to

⁴ Wikipedia organizes vast amount of human knowledge. In addition, it undergoes constant development so its breadth and depth steadily increase over time. Finally, Wikipedia contains abbreviations, neologisms, terms in slang, and domain-specific technical terms.

support other distance types, such as single-link and average-link. The following definition is used for calculating the distance between clusters.

Definition 3 (Distance between Clusters). The *distance between two clusters* C_1 and C_2 is calculated as follows: $\text{Dist}(C_1, C_2) = \max_{f_1 \in C_1, f_2 \in C_2}(\text{Sim}(f_1, f_2))$.

Examples of clusters created by our method for different feature diagrams in the mobile phones domain are listed in Figure 3. As can be seen, some of the clusters include features whose names can be considered synonyms, e.g., cluster 4, while the reasons for the emergence of other clusters can be found in their similar descendants, e.g., cluster 5, or similar ascendants, e.g., clusters 2 and 6.

Cluster 1: voice call, calls	Cluster 4: screen, display
Cluster 2: EMS, MMS, SMS	Cluster 5: media, extras
Cluster 3: High resolution, Low resolution, color, colour, basic	Cluster 6: camera, mp3, mp4

Fig. 3. Examples of clusters generated by the suggested method

4.4 Cross PL Mining

Once the feature clusters are created, cross PL mining can be conducted in three steps. First, the relationships between the clusters are analyzed, calculating the strength of each relationship with respect to the original set of feature diagrams. Then, a domain model that uses feature diagram notation is generated. Finally, local and global improvement recommendations are generated for the original input feature diagrams.

Relationship Analysis. The relationships between features in the input diagrams induce relationships between clusters. The type of relationship between clusters is the same as that between the original features. In case of “or” or “xor” (alternative) relationships in which the feature descendants are grouped into the same cluster, the type of relationship between the corresponding clusters is changed to “optional”. Note that there may be relationships of different types between the same pair of clusters, potentially in opposite directions. In order to analyze the cluster relationships, we calculate their local and global strength as follows.

Definition 4 (Local Strength). The *local strength* of a relationship of type rel from cluster C_1 to cluster C_2 is defined as the ratio between the number of PLs involved in this type of relationship and the total number of PLs whose features appear in at least one of the clusters. Formally expressed:

$$\text{Strength}_{\text{local}}(C_1, C_2, \xrightarrow{\text{rel}}) = \frac{|\{pl | \exists f_1, f_2 \in pl \text{ such that } f_1 \in C_1, f_2 \in C_2, f_1 \xrightarrow{\text{rel}} f_2 \in pl\}|}{|\{pl | \exists f \in pl \text{ such that } f \in C_1 \cup C_2\}|}$$

Definition 5 (Global Strength). The *global strength* of a relationship of type rel from cluster C_1 to cluster C_2 is defined as the ratio between the number of PLs involved in this type of relationship and the total number of PLs used as input to the method. Formally expressed:

$$Strength_{global} (C_1, C_2, \rightarrow^{rel}) = \frac{|\{pl | \exists f_1, f_2 \in pl \text{ such that } f_1 \in C_1, f_2 \in C_2, f_1 \xrightarrow{rel} f_2 \in pl\}|}{|\{pl\}|}$$

As an example, consider Figure 4. Parts (a) and (b) of this figure represent two portions of feature diagrams in the mobile phone domain (part (a) is actually taken from Figure 1). Assuming having only three input diagrams, two of which are given in this figure and the third one does not refer to basic/utility functions at all, Figure 4(c) presents a graph that results from feature clustering and relationships analysis. Note that in two cases there are both optional and mandatory relationships between the same two clusters. The local and global strengths in these cases indicate that there is one input feature diagram in which the mandatory relationship appears between corresponding features, while the optional relationship appears either explicitly (as an optional relationship) or implicitly (as “or”/“xor” relationships) in 1 or 2 input diagrams.

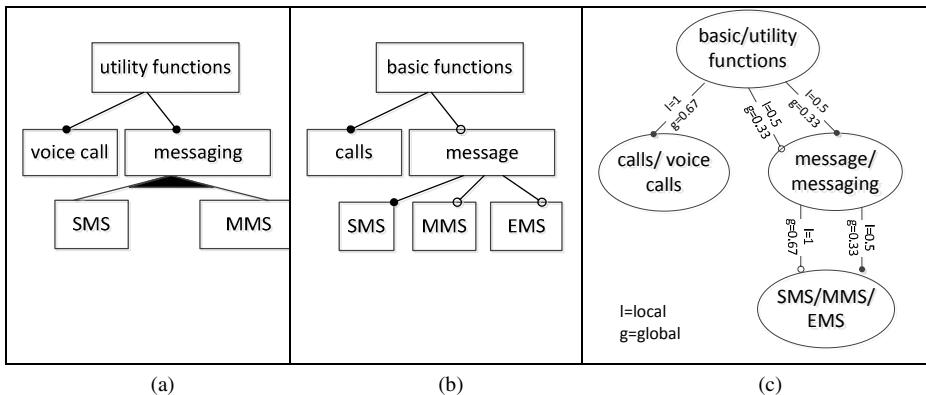


Fig. 4. An example of the outcome of the relationships analysis step: (a)+(b) parts of input feature diagrams and (c) the generated cluster level relationships

Domain Model Generation. Although the graph generated in the previous step is very informative (i.e., includes all the relationships that appear in the input feature diagrams), it is very complex. In particular, it may include cycles and several relationships between the same two clusters. Due to usability reasons and in order to benefit from existing analysis and validation techniques, we aim to represent the domain models as close as possible to feature diagrams. Thus, we apply a version of Edmonds' algorithm [7] for finding minimal spanning trees in directed graphs. The input for Edmonds' algorithm is a directed weighted graph with a single root. Our graph of clusters is actually a directed graph with local and global strengths as weights. However, we cannot guarantee that this graph will always have a single root. Thus, we examine whether the root features of the different input diagrams are grouped into a single cluster or into different clusters. In the first case, the cluster to which all the root features correspond is defined as the root of the graph. In the latter case, a pseudo cluster is added as the root. All clusters that include root features of the input

diagrams are defined as optional children of the newly added root. The global and local strengths of these edges are updated according to the fraction of the PLs in which the corresponding root features appear.

As a first step Edmonds' algorithm replaces any set of parallel edges (namely, edges between the same pair of vertices in the same direction) with a single edge. In order to determine the types of the relationships resulted after this step, the method uses Table 1. This table defines for sets of relationship types the least constraining type. The table was built based on the semantics of feature relationships: (1) being mandatory is more constraining than being optional, an alternative (in “xor” relationships), or an option in “or” relationships; (2) being an alternative is more constraining than being optional or an option in “or” relationships; and (3) being an option in “or” relationships is more constraining than being optional. The local and global strengths of the single created relationship between the two clusters are updated to reflect all the relationships it represents. For example, the local and global strengths of the relation ‘basic/utility functions’ $\xrightarrow{\text{optional}}$ ‘message/messaging’ will be respectively 1 and 0.67.

Table 1. The least constraining relationships types

Existing types of relationships	The least constraining relationship type
Only mandatory relationships	●
At least one alternative relationship (no optional and ‘or’ relationships)	● A
At least one optional relationship (no ‘or’ relationships)	● A ○
At least one ‘or’ relationship	● A ○ A

Next, Edmonds' algorithm constructs the spanning tree considering, in each iteration, the minimal-weighted edge that can be added without forming a cycle. Since the weights in our case represent strengths rather than costs, we modified the edge selection criterion: in each iteration, the relationship with the *maximal* local strength is selected⁵. In case of several maximums, the algorithm chooses the one with the maximal global strength. The resultant spanning tree is considered the domain model.

As an example for generating a domain model for mobile phones, we used nine different PLs, seven of which were taken from S.P.L.O.T – an academic feature diagrams repository [22]. The two additional ones were specifically modelled to introduce some challenges to better evaluate the method. In particular, we added synonyms and antonyms and we modelled the hierarchies of features using different nesting strategies. The number of features in the diagrams range from 10 to 25 and the

⁵ The method uses first the local strength rather than the global strength since the global strength depends on the amount of overlap among the input feature diagrams.

number of levels were 3-4. The relative simplicity of the diagrams enabled us examining the method outputs manually and conducting a throughout evaluation, as will be elaborated in Section 5. Part of the generated domain model is given in Figure 5.

Recommendations for Improvement. The generated domain model can serve as a basis for both local and global improvement recommendations: local recommendations aim to refine aspects that already exist to some extent in the PL, whereas global recommendations aim to introduce new domain-related aspects to existing PLs.

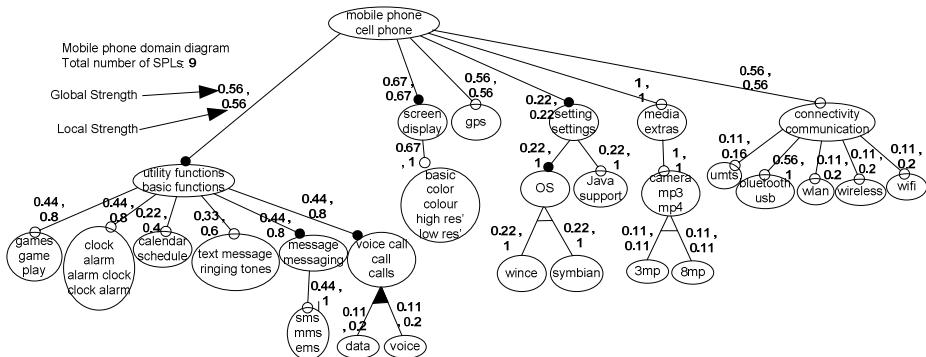


Fig. 5. A partial domain model of mobile phones

Local Recommendations. For each two related clusters whose local strength is greater than 0.5, i.e., the majority of the involved PLs include such a relationship, a local recommendation matrix (LRM) is built. The rows of this matrix are the features that belong to the parent cluster (C_1), and the columns of the matrix are the features that belong to the child cluster (C_2). However in order to avoid putting almost identical features in separate columns or rows we merge columns or rows that represent features whose similarity is very high (greater than 0.97). The LRM [i,j] cell includes a list of PLs in which a relation from f_i to f_j exists, where $f_i \in C_1$, $f_j \in C_2$.

Once the matrix has been constructed, the method searches for each column (a feature in the child cluster) PLs that do not appear in this column but do appear in other columns. For each such PL the method retrieves the row features (i.e., features in the parent cluster) with which this PL is associated. The method recommends adding the child feature under the retrieved parent features.

Examples of recommendations we got when running the method on the nine feature diagrams we used in the mobile phones domain are: (1) Consider adding the option 'mp4' under the feature 'extras' in PL #7; (2) Consider adding the mandatory feature 'messaging' under the feature 'basis functions' in PL #7; and (3) Consider adding the option 'usb' under the feature 'connectivity' in PL #8.

Currently, the method recommends only on adding features which may be required and ignores deletion or modification of existing features, assuming that the feature diagram of each PL is correct and potentially include specific aspects that are not relevant to the entire domain. Still, as future steps, we do intend to expand the types of recommendation and generate recommendations on deletion and modification of features as well as on restructuring of feature diagrams.

Global recommendations. In the domain model, relationships whose global strength is greater than 0.5 can be considered meaningful, as the majority of the input PLs exhibit them. Thus they are recommended to PLs that do not include them in the following procedure⁶. Starting from a cluster with the minimal number of incoming arcs and for each two clusters which are connected with a relationship whose global strength is greater than 0.5, a list of all PLs that appear in the parent cluster but do not appear in the child cluster is created. For each PL in the created list, the method recommends adding features from the child cluster under the feature or features from the parent cluster that are already included in the examined PL. We further virtually insert the PL on which a global recommendation is generated to the child cluster, so that the method will be able to continue analyzing the sub-tree of that child cluster and generate additional appropriate recommendations. The above procedure is repeated until the whole domain model is traversed.

Examples of global recommendations generated by our method for the mobile phones domain are: (1) Consider adding the optional feature 'connectivity'/communication' under the feature 'mobile phone' in PLs #1, #4, #5, and #6; In case you added the feature 'connectivity'/communication', you may also consider adding the optional features 'blue tooth' and/or 'usb' as sub-features; and (2) Consider adding the optional feature 'GPS' under the feature 'mobile phone' in PLs #2, #3, #7, and #9.

5 Preliminary Results

In order to evaluate the suggested method, we implemented a prototype and run it on the set of nine PLs mentioned in Section 4.4. We got 29 clusters, each of which included between 1 to 19 features. Part of the generated domain model is depicted in Figure 5. In addition, 28 recommendations for improvement were generated. Examples of such recommendations are also given in section 4.4. To check the correctness and likelihood of the generated recommendations, we created a questionnaire with 20 statements on the mobile phones domain based on these recommendations. We asked people to grade their degree of agreement with these statements. Examples of the statements we used are: (1) A mobile phone which has messaging services should have the ability to send and receive SMS; (2) A mobile phone is likely to have a GPS.

20 people were asked to fill this questionnaire. All respondents, except of one, did not work in the domain of mobile phones, but were familiar with the domain as users for 14 years on the average. The single respondent who worked in a mobile phones company worked there for only one year. Most respondents evaluated their familiarity with the domain as 'good' (11) or 'very good' (3). For each statement, the respondents were asked to mark whether they 'absolutely agree', 'partially agree', or 'disagree' with the statement. The respondents could also mark that they have no sufficient information on the subject and hence cannot decide on their degree of agreement with the statement ("don't know"). Overall, no significant difference was found between the responds of respondents with good familiarity with the domain and those with poor familiarity with the domain.

⁶ Note that such recommendation becomes meaningful as the number of used PLs increases.

The analysis of the results reveals (see Figure 6 (a)) that in 48% of the cases the respondents absolutely agreed with the specified statements and in 27% of the cases they partially agreed. The percent of disagreement is low and stands on 16%.

In order to further analyze the results, we categorized the statements into four groups: (1) *Local*: Statements that are based on local recommendations; (2) *Global*: Statements that are based on global recommendations; (3) *Other*: Statements that are not based on global and local recommendations, but on additional information (mainly "requires" and "excludes" constraints); and (4) *False*: Statements that include false recommendations, i.e., statements that were not generated by our method and in most cases represent loose relationships between clusters. For enabling aggregating the results of the FALSE category with the results in the other categories, the statements in that category were phrased negatively, namely, highly agreeing with a statement in this category confirms that it is indeed false. Figure 6(b) shows the distribution of the answers in each category. As can be seen the level of agreement with the local and global recommendations is almost the same and stands on about 77% (aggregating absolute and partial agreement). However, there is a significant difference in the level of disagreement on local and global recommendations: while in only 12% of the cases there is disagreement with the local recommendations, the level of disagreement with global recommendations is almost twice higher, namely 22%. Indeed the percentage of respondents who claimed that they have no sufficient information to decide on their level of agreement on local recommendations is 10%, which is exactly the difference in the disagreement levels on the global and local recommendations. When checking the statements on which many respondents answer "don't know", we noticed that the difference is likely to be due to the usage of advanced terms (such as Java Support and EMS) rather than to the differences between the nature of local and global recommendations.

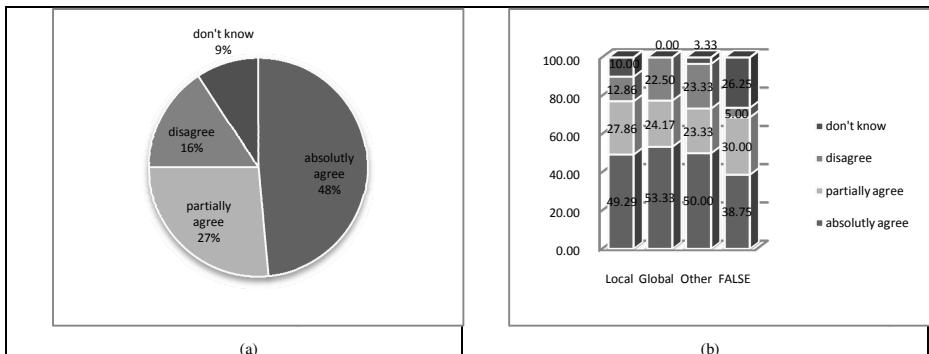


Fig. 6. Evaluation results: (a) overall, (b) per category of questions

Note that the respondents tend to claim that they do not have enough information to decide about their degree of agreement with statements in the FALSE category. Furthermore the percentage of absolute agreement on statements in this category is the lowest. These results may be explained by the greater level of exertion when judging statements which were generated from loose relationships.

6 Summary and Future Work

Domain analysis is highly important for developing PLs and particular software systems. Currently domain analysis methods focus on extracting and representing the common and variable aspects of a particular PL. While this is important, it is further essential to study the whole domain as several different PLs may exist, in the same company or in a consortium of companies that are willing to cooperate for some reasons. In this research we call for constructing the domain knowledge through cross product line analysis in order to improve the correctness and completeness of existing PL artifacts and future ones. To this end, we introduce a method that calculates feature similarity, clusters similar features, and creates domain models. The resultant domain models do not simply merge the existing artifacts, but consolidate the domain knowledge and represent it in a coherent way using a feature diagrams notation. Based on the domain models, global and local recommendations can be generated for improving the quality of existing PL artifacts in the domain. The domain models can be further used to create and validate new products, as well as to support interoperability of different, separately developed PLs. The preliminary results from evaluating the method show that the method's outputs reflect human perception of the examined domain.

Future research includes several directions. First, we intend to improve the calculation of feature similarity by examining additional techniques, and particularly those for measuring semantic similarity. Second, the types of recommendation need to be explored, to include recommendations on deletion and modification, suggest restructuring of feature diagrams, and rank the appropriateness of the generated recommendations. We may need to use besides the feature diagrams additional artifacts for this purpose, e.g., descriptions and code. Finally, additional evaluations of the method are required and especially in industrial settings taking into account different viewpoints of stakeholders. In particular, we will further compare our method to alternative approaches in order to better analyze its benefits and limitations.

References

1. Acher, M., Cleve, A., Collet, P., Merle, P., Duchien, L., Lahire, P.: Reverse engineering architectural feature models. *Software Architecture*, 220–235 (2011)
2. Acher, M., Cleve, A., Perrouin, G., Heymans, P., Vanbeneden, C., Collet, P., Lahire, P.: On extracting feature models from product descriptions. In: *Proceedings of the 6th VaMoS Workshop*, pp. 45–54. ACM Press (2012)
3. Arango, G.: Domain analysis methods. In: Horwood, E. (ed.) *Software Reusability*, Chichester, England, pp. 17–49 (1994)
4. Choi, N., Song, I.Y., Han, H.: A survey on ontology mapping. *ACM Sigmod Record* 35(3), 34–41 (2006)
5. Clements, P., Northrop, L.: *Software Product Lines: Practices and Patterns*. Addison-Wesley (2001)
6. Czarnecki, K.: Generative Programming: Methods, Techniques, and Applications Tutorial Abstract. In: Gacek, C. (ed.) *ICSR-7. LNCS*, vol. 2319, pp. 351–352. Springer, Heidelberg (2002)

7. Edmonds, J.: Optimum branchings. *Journal of Research of the National Bureau of Standards – B: Mathematics and Mathematical Physics* 71B(4), 233–240 (1967)
8. Frakes, W.B., Kang, K.: Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering* 31(7), 529–536 (2005)
9. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55, 119–139 (1997)
10. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence* 14, 771–780 (1999)
11. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 1606–1611 (2007)
12. Giachetti, G., Marín, B., Pastor, O.: Using UML as a Domain-Specific Modeling Language: A Proposal for Automatic Generation of UML Profiles. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) *CAiSE 2009. LNCS*, vol. 5565, pp. 110–124. Springer, Heidelberg (2009)
13. Kamvar, S.D., Klein, D., Manning, C.D.: Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In: *Proceedings of 19th International Conference on Machine Learning*, pp. 283–290 (2002)
14. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study, SEI Technical Report (1990)
15. Kurita, T.: An efficient agglomerative clustering algorithm using a heap. *Pattern Recognition* 24(3), 205–209 (1991)
16. Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., Zave, P.: Matching and Merging of Statecharts Specifications. In: *29th International Conference on Software Engineering, ICSE 2007*, pp. 54–64 (2007)
17. Rabkin, A., Katz, R.: Static extraction of program configuration options. In: *33rd International Conference on Software Engineering (ICSE)*, pp. 131–140 (2011)
18. Rasmussen, E.: Clustering algorithms. *Information Retrieval: Data Structures and Algorithms*, 419–442 (1992)
19. Reinhartz-Berger, I.: Towards Automatization of Domain Modeling. *Data & Knowledge Engineering* 69, 491–515 (2010)
20. She, S., Lotufo, R., Berger, T., Wasowski, A., Czarnecki, K.: Reverse engineering feature models. In: *33rd International Conference on Software Engineering, ICSE*, pp. 461–470 (2011)
21. Shvaiko, P., Euzenat, J.: Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering* 25(1), 158–176 (2013)
22. S.P.L.O.T Software Product Lines Online Tools, <http://www.splot-research.org/>
23. Talukdar, P.P., Ives, Z.G., Pereira, F.: Automatically incorporating new sources in keyword search-based data integration. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pp. 387–398 (2010)
24. Van Deursen, A., Klint, P.: Domain-specific language design requires feature descriptions. *Journal of Computing and Information Technology* 10(1), 1–17 (2004)
25. Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-based integration of information – a survey of existing approaches. In: *IJCAI 2001 Workshop: Ontologies and Information Sharing*, pp. 108–117 (2001)
26. Weston, N., Chitchyan, R., Rashid, A.: A framework for constructing semantically composable feature models from natural language requirements. In: *Proceedings of the 13th International Software Product Line Conference, SPLC 2009*, pp. 211–220 (2009)

27. Wikipedia. Mergers and acquisitions,
http://en.wikipedia.org/wiki/Mergers_and_acquisitions
28. WordNet: a lexical database for English, <http://wordnet.princeton.edu/>
29. Wu, Z., Palmer, M.: In Verbs semantics and lexical selection. In: Association for Computational Linguistics, pp. 133–138 (1994)
30. Wulf-Hadash, O., Reinhartz-Berger, I.: Cross Product Line Feature Analysis. In: Proceedings of the 7th VaMoS Workshop, pp. 123–130. ACM Press (2013)

Incremental Method Enactment for Computer Aided Software Engineering Tools

Kevin Vlaanderen, Geurt van Tuijl, Sjaak Brinkkemper, and Slinger Jansen

Utrecht University,
Department of Information and Computing Sciences,
P.O. Box 80.007,
3508 TA, Utrecht, The Netherlands

{k.vlaanderen,g.j.vantuijl,s.brinkkemper,s.jansen}@uu.nl

Abstract. In most cases, enactment is the most resource consuming aspect of process improvement, as large process changes are put into practice. Problems that typically are encountered include ineffective process changes, resistance from employees, and unclarity about the advantages of the new process. These problems can be avoided by incrementally implementing process changes, especially if the enactment is supported by process management tools that immediately change the processes and workflow in information systems. In this paper, we explain and demonstrate the concept of incremental method enactment for CASE tools. The concept is evaluated through a prototype, which is assessed by industry experts. The results of this study point give direction towards the further development of incremental method enactment.

Keywords: Method Engineering, Method Evolution, CASE tools, CAME tools, Enactment.

1 Introduction

The methods that organizations employ are subject to constant change. Implementing changes to existing processes requires a lot of time and effort, and its success is prone to many factors, especially when the changes are significant. Issues such as resistance to change, lack of evidence, and resource constraints cause many process improvement efforts to fail [2]. Not surprisingly, this implementation or enactment of processes and process changes is an import subject within the Software Process Improvement domain [10,11,4].

Much research has been performed related to determining *what* changes should be introduced in order to improve process, and thus software, quality. This has resulted in a wide range of frameworks including CMMI [6], and SPICE [9]. Such frameworks need the addition of research on *how* changes should be introduced. The success of process improvement is highly dependent on contextual factors that go beyond the improvement content. In their analysis of SPI success in small to medium enterprises, [20] have determined multiple success factors, two of which are particularly important in the context of this research:

- ”Guide the improvement programme, following a systematic and coherent initiative by means of specific procedures and combining different approaches. This procedure must follow an iterative and incremental approach (prioritize the improvement points defined by the organization) that allows a continuous adoption of improvement practices.” [20]
- ”Tackle the problem of improvement from the technical perspective.” [20]

The first of these two factors or guidelines lays at the core of what is called *incremental method evolution* or incremental process improvement, which deals with the step-wise improvement of process development processes [26]. Incremental method evolution focuses on delivering tools and techniques that enable small, local changes to a method with the goal to minimize the overall disturbances on a business process and to allow for an iterative and ‘natural’ approach to process improvement. The second guideline is important because, with the advent of modern technology, automated process and computer-supported process execution becomes more and more feasible.

Most software organizations rely on several CASE tools during the development of software products. Such tooling often requires significant configuration and maintenance during its lifecycle, and adaptations to it can thus cause significant resistance among system administrators and users alike. Incremental method enactment as we use it here, aids the transformation of a method and the implemented process by alleviating some of the issues that come in play when they are changed, mainly with regard to tooling maintenance.

These challenges form the basis for the research question answered in this paper, which is formulated as *How can the enactment of incremental method changes in CASE tools be facilitated?*

The research presented in this paper elaborates on the enactment of changes in situational methods, from a technical perspective, in a dynamic environment. We propose a mechanism for synchronizing the evolution of situational methods and the tools supporting them throughout the lifecycle of a method. This solution, referred to as incremental method enactment, describes the process of automatic changes in software engineering tools according to incremental changes in methods and their instances. The mechanism is demonstrated by a proof of concept. Process Deliverable Diagrams are used for modeling multiple method increments from a process and work product perspective. These diagrams are translated automatically into the datamodel of a CASE tool in order to update both the workflow of the tool as well as the work products facilitated by it. The proposal is validated through multiple interviews with industry experts.

2 Incremental Method Enactment

The research community has long recognized that the idea of fully standardized and stable methods across multiple organizations is a utopia [5,19]. Organizations employ multiple, often interdependent methods that vary in their amount of formalization and quality. The environment in which the organizations operate changes, and the internal methods need to change accordingly. Existing

information systems development methods are often too generic and cannot be followed [15]. They need to be tuned to a specific situation for the project at hand [14,22,3] by incorporating the uniqueness of a specific project, which lies at the core of situational method engineering (SME). SME denotes a group of techniques that allow organizations to develop tailor made methods tuned to their specific situation and is defined as the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems [3]. During the last several years, several modularization constructs have been proposed for situational method engineering. Although these constructs have many aspects in common, some essential differences exist. Extensive comparisons of these constructs were performed by [7], [1] and [8]. Each of these techniques focuses on the construction and/or adaptation of methods based on (parts of) previous methods. Instead of spending effort on adapting the organization to a standardized process, it is spent on thoroughly analyzing the organization and developing a process that better fits the organizational structure, capabilities, and habits. Theory predicts that the implementation of such situational methods has a higher chance of success by lowering the barrier for introducing changes.

The concept method enactment, depending on the research line, has gained much attention in the software process improvement (SPI) and situational method engineering (SME) domains. In general, enactment refers to the instantiation of a process model or a method to a real-life scenario, in other words 'putting it into action'. As was already recognized by [10], enactment deals not only with the implementation of a method through tooling, but also with training, controlling, etc. of human actors that perform the process. From the viewpoint of situational method engineering, method enactment represents the final stage of the construction or the tailoring of a method for a specific project at hand. It completes a round-trip, starting from an analysis of the current situation and ending at an improvement of that situation.

The Structure of Method Increments. Incremental method enactment implies an iterative approach to the development and instantiation of a process. As the term implies, it is based on the notion of a method increment. The basic components of a method increment are generally a description of the changes involved, both in terms of activities as well as deliverables, and possibly a description of the goal and experiences. The latter part is often described as the rationale. Based on the notion of design rationale, [21] informally define method rationale as information about decisions that lead to a certain meta-model. From a more structural viewpoint, a method increment has been defined as a collection of method fragments that have been introduced in a method between two points in time [29]. Conceptually, it represents a coherent set of changes to an existing method in order to facilitate evolutionary changes.

Continuing in this line, we adopt the notion of method increments and its proposed modeling approach. Essentially, method increments are modeled in the form of Process-Deliverable Diagrams (PDDs) [28], where each consecutive

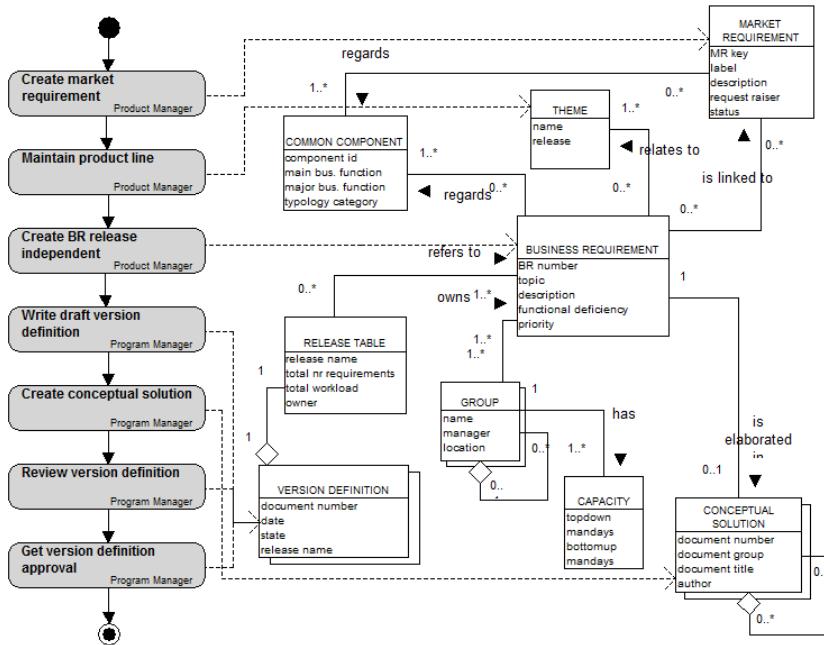


Fig. 1. Example Process Deliverable Diagram; Increment 4 of the Case Study

model describes the state of a method fragment at a specific point in time [29]. The evolution of a method fragment is therefore described by a series of PDDs, thus creating something that resembles a storyline.

In the context of method engineering, and described in terms of the Meta Object Facility (MOF) levels [12], the process part of a PDD is at level M1 (class level), while the product part remains one level higher at M2 (metaclass level), as shown by [17]. During the execution of a process, both parts are instantiated, resulting in specific activities (M0 level) and models such as requirements or use case diagrams (M1 level). CASE tools are designed to support the instantiations, thus acting at the same meta-levels as a PDD. Figure 1 demonstrates this by showing one of the PDDs that have been used during this study. This specific example describes the process of capturing market requirements and transforming them into business requirements that end up in a specific release.

2.1 The Process of Incremental Method Enactment

Both the meta-modeling activity as well as the modeling activity can be performed in a variety of ways and supported by a variety of tools, depending on the specific context and processes. For instance, numerous tools have been developed for supporting the requirements engineering activity. Each tool provides more or less freedom to the user in terms of modeling paradigms that can be used. On the other hand, the meta-modeling activity can be supported by a variety of

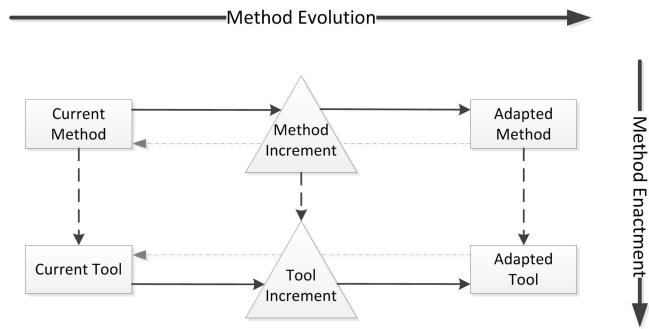


Fig. 2. Conceptual Illustration of Incremental Method Enactment

Computer Aided Method Engineering (CAME) tools, supporting activities such as the creation, modification, and combination of method fragments. In some cases, such as in the case of MetaEdit+ [18], both worlds are combined, allowing method engineers to define a modeling environment that can be instantiated and in turn be used by, for example, the software engineer. This by itself can be seen as a form of method instantiation, as it involves the transformation of the meta-model level to the model-level. However, true situational method engineering solutions should not force engineers (on either level) to specific tools. A typical software engineering process often involves multiple models and is subject to the capabilities and the context of the organization.

Based on the above, we can say that incremental method enactment, from a technical perspective, is a continuous adaptation of the meta-model combined with a continuous synchronization of the meta-model and the CASE tool, as illustrated by Figure 2. The goal is to facilitate changes in the software engineer's process by correctly transforming a CASE tool's configuration to match the method's meta-model. Changes to the activity side result in an altered workflow, while changes to the deliverable side result in altered work products. The challenge is then to support the enactment of both the process as well as the deliverable aspect of a method fragment while taking into account the specific organizational context.

This can be done through an intermediary tool, such as demonstrated in Figure 3. Such a tool extends the functionality of the CAME environment by linking the meta-meta-model (i.e. the modeling language used for creating method fragments) to a specific CASE tool's interface. The constructs used within the instantiation of this meta-meta-model are translated into the constructs that are used within the CASE tool. On each consecutive change to the method, the exact adaptations are calculated and propagated to the CASE tool, enabling a smooth transitions from one method version to the next.

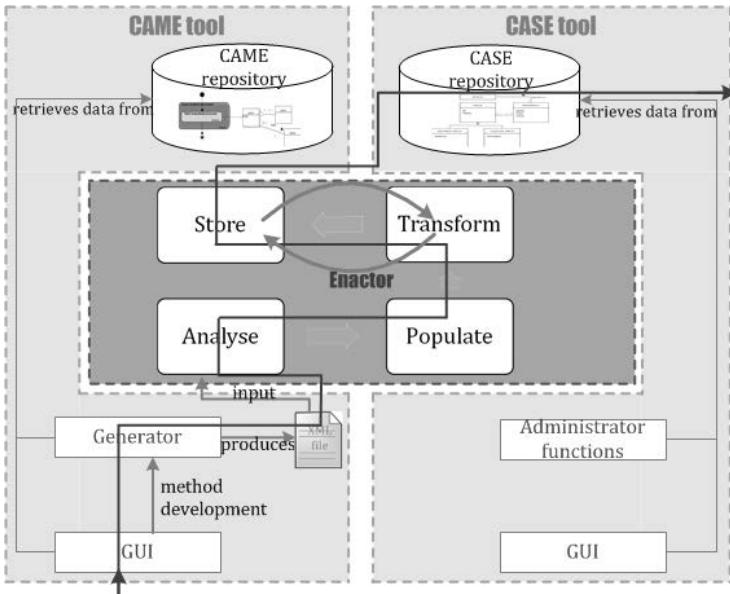


Fig. 3. Overview of the Enactment Mechanism

Such a translation is highly dependent on both modeling environments, and defining a modeling paradigm agnostic approach lies beyond the scope of this paper. However, we demonstrate and evaluate the potential workings of such an approach through a prototype in the next chapter.

3 Prototype

To demonstrate the incremental method enactment approach, we have built a prototype enactment mechanism. This mechanism is used to evolve a live CASE tool according to method changes. The prototype allows us to evaluate both the requirements for implementation as well as the mechanism itself.

3.1 Selection of Tools

On the one hand, the CAME tool needs to satisfy several technical requirements. The CAME tool should be able to create valid increments according to the chosen modeling paradigm, in this case the PDD. Another requirement is that the CAME tool is capable of transforming a graphical representation to structured textual data. Finally, the CAME tool should include the possibility of invoking the enactment mechanism that is responsible for pushing the changes to the CASE tool.

Through an analysis of previous literature, we have identified seven available CAME tools, namely MERET [16], MethodBase [23], Decamerone [14],

MENTOR [24], MetaEdit+ [18], MERU [13] and Method Editor [22]. Out of these tools, MetaEdit+ is the most actively developed. In earlier research, a metamodel has been created for Process-Deliverable Diagrams in MetaEdit+ (see [27]). This metamodel is used to create valid PDDs through the MetaEdit+ interface. Moreover, MetaEdit+ is able to transform a graphical fragment to structured text and allows for the execution of external functionality.

Similarly, the CASE tool is subject to a set of high-level requirements as well. Primarily, the CASE tool needs to include a means to support the process side as well as the deliverable side of a PDD. This can be translated to the ability of workflow management in addition to work product management. Furthermore, the configuration of the CASE tool should be accessible and modifiable through an external interface. This interface can be provided in terms of an Application Programming Interface (API), a manual function such as the ability to import external configuration files, or direct access to the application database. In addition, the CASE tool needs to be rather mature. CASE tools that were too superficial or too concise (such as small mock-ups or simple programming classes) were eliminated from the list. Finally, The selected CASE tool should be well-known to the IT-industry. We marked a CASE tool as 'well-known' if we found several prominent companies (i.e. customers of the CASE tool) that are using the CASE tool.

For the selection of an appropriate CASE tool, we have created a list of available CASE tools, based on manual research on the internet. In total, we found over 50 usable CASE tools. After the selection procedure, we chose to use Jira¹ during the development of the prototype. Jira supports the definition of detailed workflows, enabling the implementation of the activity side of PDDs. The tool is flexible and very configurable. In addition, it is a widely used tool, enhancing the practical relevance of this research.

3.2 Mapping Increments to the CASE Tool's Configuration

After selecting the appropriate tools, we created a mapping that allows us to convert the data of a method fragment to the CASE tool's configuration model. Figure 4 depicts this mapping.

On the left-hand side of Figure 4, a simplified fragment in the form of a PDD is shown, encircled with a dotted line. On the right-hand side of Figure 4, the essential elements of the CASE tool's datamodel are shown. The lines link the method fragment meta-model to the datamodel. Table 1 provides a textual explanation of the relationship between both fragments.

The diagram name of the fragment that a user creates is used to create an issue type in Jira. For instance, if you create a first version of a fragment that is entitled 'Requirement management process', this diagram name will be used to create an issue type in Jira. All the data that is entered in Jira underneath this issue type belongs to that specific fragment. When an updated version of the fragment is created, the existing issue type will not be affected. Issue types can be used in different projects or in one project only.

¹ <http://www.atlassian.com/software/jira/>

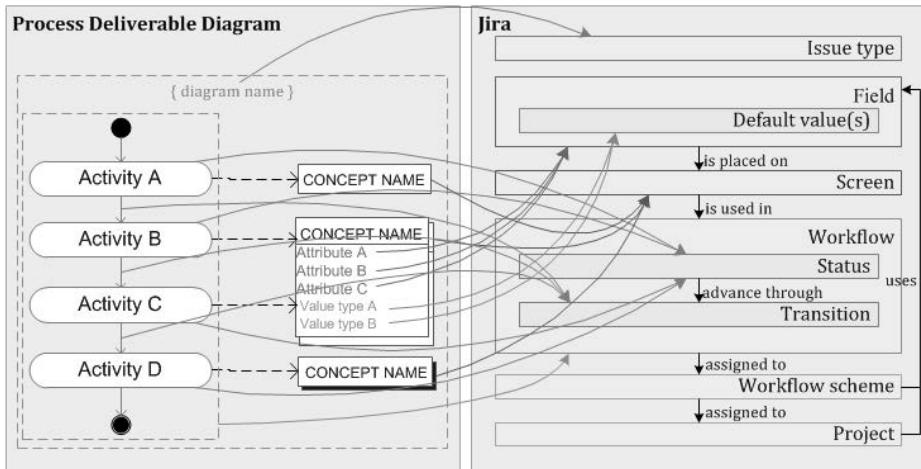


Fig. 4. Mapping of PDD and the CASE tool datamodel

The complete process side of the fragment is used to create a workflow in Jira. A workflow in Jira is only represented by a name and the entire workflow is based on an XML structure. A workflow in Jira is similar to the process side that is modeled in a PDD. A workflow exists of a status and transitions. A status is a step in the workflow through which you advance while working in Jira. For instance, if you complete activity A in the workflow and mark this as complete, you advance to the next status (i.e. the next activity). Advancing through statuses in Jira requires transitions. Transitions are moments in the workflow in which you can complete one activity and continue with the next activity.

Table 1. Mapping of Method Fragments to Jira

#	Fragment	Relationship	Jira tool	fragment	Usage domain
1	Diagram name	is used to create an	Issue type		per project
2	Process side	will produce a	Workflow		per project
3	Activity	is used to create a	Status		in a workflow
4	Transition	is used to create a	Transition		in a workflow
5	Concept	is used to create a	Screen		per project
6	Attribute	is used to create a	Field		per project
7	Value type	will be used as (a)	Default value(s)		per field

An activity within a PDD is used to create a status in Jira. The status in Jira is a direct copy of the activity as described in the PDD. Statuses in Jira are used within workflows. They are used to indicate at which point the user currently

is in the entire workflow, similar to a PDD. Each status is linked to a screen so that a status can present information to the user.

A transition in a PDD is the same as a transition in a workflow in Jira. They are used to close an activity a user is currently dealing with and to continue with the next activity in the workflow. The name of the transition is based on the upcoming activity. For example, if you are at 'Gather requirements' in the workflow, the transition name will be 'Write draft version definition' if this is the upcoming activity.

The name of a concept is used to create a screen in Jira. For instance, if you have a concept called REQUIREMENT, a screen is created with the name REQUIREMENT. A screen contains fields that are based on attributes of the concept. In the event that a concept does not contain attributes, the screen does not have any input fields. Although this seems to make little sense, the screen is linked to a status that can contain many other functions such as attaching a document or adding comments to a particular status. In that case having a screen can still be very useful even though it contains no fields.

An attribute of a concept within a PDD is used to create a field with the same name. The field that is created will be placed on the screen of the respective concept. For instance, if a concept called REQUIREMENT has an attribute called code, a screen is created (namely REQUIREMENT) which has a field called code. If an attribute has attributes, these attributes will be used as default values of the respective field. For instance, if the attribute code has attributes such as G1, G2 or G3, a select box field is created that contains the values G1, G2 or G3. These values can be selected from a drop down box for instance.

3.3 The Enactment Mechanism

The enactment mechanism facilitates the translation from a PDD to the internal datamodel of our CASE tool. After a fragment has been created in MetaEdit+ (see Fig. 3 for an example), a MERL script (an internal scripting language) exports the fragment to an external tool. This tool extracts the relevant meta-data from the fragment, including the fragment name and its version, and converts the fragment to an intermediate structure based on the PDD meta-model as defined by [29].

Once all relevant data has been extracted, it is shown in a user friendly manner, in order to let the user verify the consistency of the data. Once the data is verified, the enactment process is triggered. For each relevant database table of the CASE tool, three objects are created based on the Model-View-Controller (MVC) design pattern; a Value Object (VO), a Data Access Object (DAO), and a Controller Object (CO). These objects form the connection between the MetaEdit+ datastructure and the CASE tool datastructure.

Through a set of prepared SQL statements, the data fragment is then stored in the CASE tool configuration database. Prepared statements are used to increase safety and avoid SQL injection (i.e. attacking the database deliberately through

query manipulation). We used PDO in conjunction with MySQL 5.0 because this allows us to abort the entire querying execution process when a single query fails. In the case a query fails at any point in the query execution process, the whole query execution process can be rolled back without leaving any data behind. In that case not a single change is made to the data in the database of the CASE tool, avoiding contamination and perhaps a broken system.

of each fragment can be found in [30].

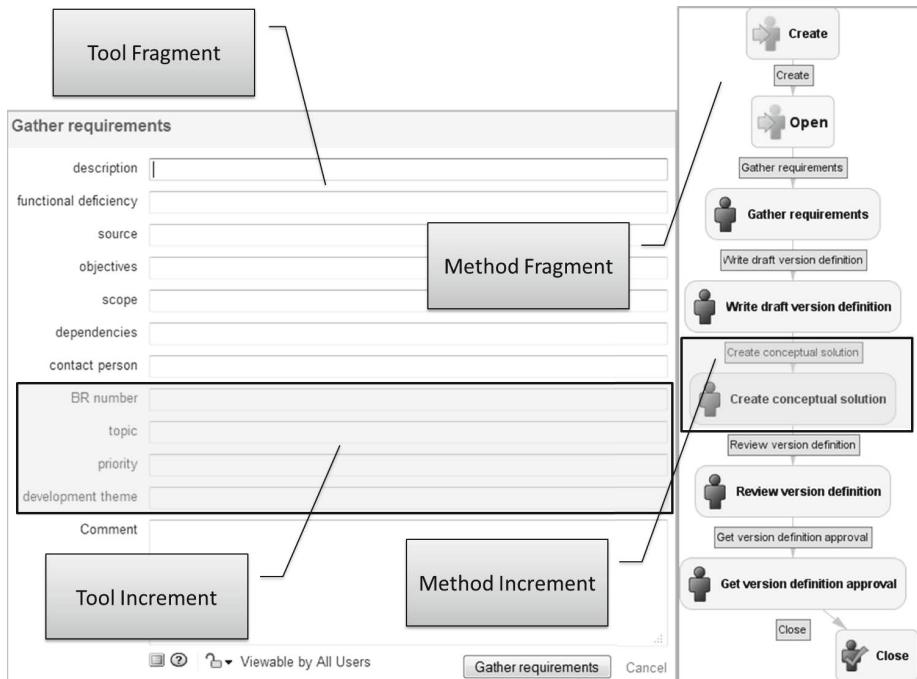


Fig. 5. Adapted Method Fragment in the CASE tool.

Figure 5 shows the method increment and tool increment based on increment #4 in [30]. On the left side of Figure 5, the fields for the respective concepts of increment #4 are shown. The fields in Jira are based on the attributes of the concepts REQUIREMENTS DOCUMENT and REQUIREMENT. On the right hand side of Figure 5, the workflow is shown that is in accordance with the workflow in fragment #4. The shaded areas denote the method and tool increment.

4 Evaluation

The research presented in this paper demonstrates an early attempt at supporting the technical enactment of continuous method improvements. To determine

which aspects should receive more attention in future research, we used the prototype to evaluate the approach. In this section, we summarize both a technical evaluation as well as an expert evaluation.

4.1 Technical Evaluation

We modeled five different increments that we used to evaluate the enactment mechanism. To emphasize the notion of mimicking a real-life setting, we selected the base method and the first four method increments from a previously conducted case study [30] at a vendor of ERP software. The increments show the evolution of a requirement management method that advances from a very simple method to a more elaborate approach. The increments, along with a textual elaboration of each increment, can be found in [30]. An overview of the method increments used during this study is found in table 2.

Table 2. Overview of Method Increments

# Goal	Date
0 Introduction requirements document	1994
1 Introduction design document	1996
2 Introduction version definition	1998, May
3 Introduction conceptual solution	1998, November
4 Introduction requirements database, division market and business requirements, and introduction of product families	1999, May

During the development of the prototype, we encountered some issues that still inhibit a fully automated approach to method enactment.

Standardization of Process Deliverable Diagrams. In principle, the PDD format has been clearly defined and formalized [28]. However, in practice we encountered many PDDs with different appearances and rules. Unfortunately, unambiguous modeling rules and notations still lack which often caused trouble when automating the data of a PDD through a mechanism.

Usage of Open, Closed and Sub Activities. Open, closed and subactivities in a fragment indicate different levels of aggregation within the process. However, the ability to implement these aggregation levels highly depends on the CASE tool, and many don't distinguish between different levels of activities, e.g. phases, steps; each activity forms a step in the workflow. Because the work-flow possibilities in Jira lack comprehensive support, each activity in a fragment is seen as a regular step in the workflow.

Usage of Forks, Joins and Decisions. The use of forks, joins and decisions in a CASE tool is limited, caused by the amount of default features that are available in a CASE tool. In our case, we could use decisions and/or logical flows in our fragment only because Jira lacks comprehensive support for these special types of activity nodes.

Generalizations, Associations and Aggregations. In our mapping, we did not find a way through which we could make sound use of generalizations, associations and aggregations. The implementation of these types of relationships is unknown, also caused by the available amount of features in the CASE tool. Jira, in its current form, was not able to provide sufficient support for these types of special relationships.

4.2 Expert Evaluation

We conducted 5 semi-structured interviews at different medium/large-scale organizations. We interviewed 5 product managers from different IT branches in order to evaluate the rationale behind the approach, the feasibility of the enactment mechanism, and the required adjustments. We posed a total of 14 questions. 7 of these were aimed at eliciting information directly related to the companies' current situation. The other 7 focused on the evaluation of the enactment mechanism. In between, a demo was shown to the product manager in the form of a video that showed the mechanism in action². During the video, a verbal explanation was provided to make things clearer to the product manager. At the end of each interview, time was available to document any kind of other information that was provided by the product manager, i.e. remaining concerns and/or opinions.

Each of the interviewees agreed on the fact that process adaptation is an important and continuous activity throughout the organization or department. Each organization struggled in the beginning with their business process(es), and each company reinvented the wheel numerous times before a standard process was in place. 4 out of 5 companies indeed had process models in place, mostly at a high level of detail. One company did not have any model in place due to its small size. Over time, each company shifted from having a very detailed process model to a more high-level process.

As indicated by the interviewees, one of the major issues with the current approach is its rigidity. A successful enactment mechanism needs to be flexible, supporting non-linear, complex methods, and allowing exceptions. This includes the support for decisions, multiple stakeholders, and a flexible stance towards the relation between the process model and reality. Most interviewees agreed that any enactment approach that forces people to work in a specific way would be met with much resistance, which is a result that is in line with most recent SPI [2,25].

² The video can be downloaded from

http://www.staff.science.uu.nl/~vlaan107/enactment_demo.wmv

Besides issues of resistance and rigidity, the interviewees commented on the fact that the current approach does not deal with all possible process model changes. It focuses mainly on the addition of steps and/or deliverables, while leaving out the deletion of process elements. This caused concern related to the reliability of the approach and the availability of data.

5 Conclusions and Further Research

In this research, we attempted to provide an answer to the following research question:

How can method increments be employed in software engineering tools and allow for incremental method enactment?

We established an alternative viewpoint on how process models can be useful within organizations. The making of process improvement models tends to absorb a lot of time and resources while they are rarely used. In this research, we demonstrated how such models can facilitate the enactment of processes by using them to adjust CASE tools.

The enactment mechanism that we developed analyzes and validates method fragments. It extracts all relevant knowledge, which is then stored in a structured manner, after which the data is converted to a standard that is compliant with the CASE tool. Finally, all the elicited data from the fragment is stored in the repository of the CASE tool. When the four phases are executed successfully in a chronological order, the configuration of the corresponding software engineering tool is automatically tailored to the method fragment. When the enactment mechanism is invoked after each created fragment, incremental method enactment is achievable in the corresponding CASE tool.

Although we used a prototype to demonstrate and evaluate the mechanism and the concepts behind it, it should be noted that the implementation was specific to Jira. Still, the mechanism was perceived as potentially useful by practitioners, and it is a first step towards enactment support for process evolution. However, the current mechanism was constrained by using a linear process flow only. In order for the mechanism to become a useful tool, the current mechanism needs more functionality by incorporating workflow support for iterative processes, decisions, joins and forks, although this is partially influenced by the selected CASE tool. The selected CASE tool should at least provide support for the process side (i.e. workflow support) and features to operationalize the deliverable side of a fragment.

The interviewees placed strong emphasis on ease of use, advanced workflow support (including decisions, multiple roles, and iterative processes), and the ability to show the transparency of (internal) processes towards the customer(s) using the approach. Therefore, during next iterations of the design process, we will focus on these aspects.

References

1. Ågerfalk, P.J., Brinkkemper, S., Gonzalez-Perez, C., Henderson-Sellers, B., Karlsson, F., Kelly, S., Ralyté, J.: Modularization Constructs in Method Engineering: Towards Common Ground? Situational Method Engineering: Fundamentals and Experiences 244, 359–368 (2007)
2. Baddoo, N.: De-motivators for software process improvement: an analysis of practitioners' views. Journal of Systems and Software 66, 23–33 (2003)
3. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Information and Software Technology 38(4), 275–280 (1996)
4. vom Brocke, J., Sinnl, T.: Culture in business process management: a literature review. Business Process Management Journal 17(2), 357–378 (2011)
5. Brooks, F.: No Silver Bullet. IEEE Computer 20(4), 10–19 (1987)
6. CMMI Product Team: Capability Maturity Model Integration (CMMI). Tech. Rep. December 2001, Carnegie Mellon Software Engineering Institute, Pittsburgh (2002)
7. Cossentino, M., Gaglio, S., Henderson-Sellers, B., Seidita, V.: A metamodeling-based approach for method fragment comparison. In: Proceedings of the 11th International Workshop on Exploring Modeling Methods in Systems Analysis and Design, EMMSAD 2006 (2006)
8. Deneckère, R., Iacovelli, A., Kornyshova, E., Souveyet, C.: From Method Fragments to Method Services. In: Halpin, T., Proper, H., Krogstie, J. (eds.) Proceedings of EMMSAD 2008, Montpellier, France (2008)
9. El Emam, K.: SPICE: The theory and practice of software process improvement and capability determination. IEEE Computer Society Press, Los Alamitos (1997)
10. Feiler, P., Humphrey, W.: Software process development and enactment: Concepts and definitions. In: International Conference on the Software Process. Software Engineering Institute, Pittsburgh (1993)
11. Gonzalez-Perez, C., Henderson-Sellers, B.: A work product pool approach to methodology specification and enactment. Journal of Systems and Software 81(8), 1288–1305 (2008)
12. Group, Object Management: Meta Object Facility (MOF TM). Tech. rep., Object Management Group (2011)
13. Gupta, D., Prakash, N.: Engineering Methods from Method Requirements Specifications. Requirements Engineering 6(3), 135–160 (2001)
14. Harmsen, F., Brinkkemper, S.: Design and implementation of a method base management system for a situational CASE environment. In: Proceedings of the Second Asia-Pacific Software Engineering Conference, APSEC 1995, p. 430. IEEE Computer Society, Washington, DC (1995)
15. Harmsen, F., Brinkkemper, S., Oei, J.L.H.: Situational method engineering for informational system project approaches. In: Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle, pp. 169–194. Elsevier Science Inc., New York (1994)
16. Heym, M., Osterle, H.: A semantic data model for methodology engineering. In: Proceedings of the Fifth International Workshop on Computer-Aided Software Engineering, pp. 142–155. IEEE (1992)
17. Jeusfeld, M.A.: A Deductive View on Process-Data Diagrams. In: Ralyté, J., Mirbel, I., Deneckère, R. (eds.) ME 2011. IFIP AICT, vol. 351, pp. 123–137. Springer, Heidelberg (2011)

18. Kelly, S., Lyytinen, K., Rossi, M.: MetaEdit+ A fully configurable Multi-User and Multi-tool CASE and CAME Environment. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) CAiSE 1996. LNCS, vol. 1080, pp. 1–21. Springer, Heidelberg (1996)
19. Malouin, J., Landry, M.: The mirage of universal methods in systems design. *Journal of Applied Systems Analysis* (1983)
20. Pino, F.J., García, F., Piattini, M.: Software process improvement in small and medium software enterprises: a systematic review. *Software Quality Journal* 16(2), 237–261 (2008)
21. Rossi, M., Tolvanen, J.: Method rationale in method engineering. In: Proceedings of the Hawaii International Conference on System Sciences, pp. 1–10. Hawaii (2000)
22. Saeki, M.: CAME: The first step to automated method engineering. In: Crocker, R., Steele Jr., G.L. (eds.) Proceedings of the Workshop on Process Engineering for Object-Oriented and Component-Based Development, pp. 7–18. ACM, Anaheim (2003)
23. Saeki, M., Iguchi, K.: A meta-model for representing software specification & design methods. In: Proceedings of the IFIP WG8.1 Working Conference on Information System Development Process (1993)
24. Si-Said, S., Rolland, C., Grosz, G.: MENTOR: a computer aided requirements engineering environment. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) CAiSE 1996. LNCS, vol. 1080, pp. 22–43. Springer, Heidelberg (1996)
25. Sulayman, M., Urquhart, C., Mendes, E., Seidel, S.: Software process improvement success factors for small and medium Web companies: A qualitative study. *Information and Software Technology* 54(5), 479–500 (2012)
26. Tolvanen, J.P.: Incremental method engineering with modeling tools: theoretical principles and empirical evidence. University of Jyväskylä (1998)
27. Vlaanderen, K., van de Weerd, I., Brinkkemper, S.: On the Design of a Knowledge Management System for Incremental Process Improvement for Software Product Management. *International Journal of Information System Modelling and Design (Selected papers of ME 2011)* (2012) (accepted for publication)
28. van de Weerd, I., Brinkkemper, S.: Meta-modeling for situational analysis and design methods. In: *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*, ch. III, p. 35. Information Science Publishing (2008)
29. van de Weerd, I., Brinkkemper, S., Versendaal, J.: Concepts for Incremental Method Evolution: Empirical Exploration and Validation in Requirements Management. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 469–484. Springer, Heidelberg (2007)
30. van de Weerd, I., Brinkkemper, S., Versendaal, J.: Incremental method evolution in global software product management: A retrospective case study. *Information and Software Technology* 52(7), 720–732 (2010)

Gamification to Support the Run Time Planning Process in Adaptive Case Management

Danny Oldenhove^{1,2}, Stijn Hoppenbrouwers^{2,3},
Theo van der Weide², and Remco Lagarde¹

¹ Atos, The Netherlands

{danny.oldenhove,remco.lagarde}@atos.net

² Radboud University Nijmegen, The Netherlands
tvdw@cs.ru.nl

³ HAN University of Applied Sciences, The Netherlands
stijn.hoppenbrouwers@han.nl

Abstract. Adaptive Case Management is used to manage unpredictable processes. These processes are mostly knowledge oriented and different roles need to collaborate to carefully plan the next steps during the execution of a case. These next steps cannot always be planned ahead, but depend on events and changes and differ for each instance. During the execution period the actual model of the run time planning, of a particular instance of a case, is made. For different roles to easily plan the correct next steps, it is important that such a case can be conceptualized and communicated. In this paper we suggest the idea of using game elements, or *Gamification*, to enhance the planning process during the execution of a case. With the use of Gamification we hope to make this process more recognizable for people and create better involvement by engaging the familiarity of games. The use of role-playing games is already being used for workshops and requirements elicitation. By building on existing work in Adaptive Case Management and Gamification we show that most games and the planning process of a case are in some respects similar. More in particular, we will discuss how we can learn from games to improve the team play during the planning process of a case. Finally this idea will be explained through an example of a planning process for an unpredictable case.

Keywords: *Adaptive Case Management, Gamification, Modeling in Run Time, Communication, Games.*

1 Introduction

This paper will focus on unpredictable processes and how specialists can be supported in their jobs to discuss and model the progress of such a process. A good example of an unpredictable processes is that of a patient in a hospital. It is hard to predict how a patient in a hospital will be treated. Only during the actual treatment of the patient, next steps will become applicable and available. Several specialists discuss the progress of the treatment plan and together plan

next steps accordingly. In fact they are modeling the actual process during the execution of such a *case*. During a modeling process, communication is vital [1]. The participants engage in communication to create an 'agreed model' [2]. During such modeling, the *process* is of paramount importance [3]. A sound process is key to understand and improve the quality of modeling [4]. With a clear goal in mind, remodeling the planning to involve next steps in the handling of a case, participants need to work together in a form of *team play*. In this paper we suggest the idea of using game elements to improve this modeling process. The use of game elements in a non-game context, or *Gamification* [5], is not a new phenomenon. Hoppenbrouwers et al. [3] discussed how Gamification can be used to improve the quality of modeling of a method or tool. In this paper we will discuss how Gamification can be used to improve the *run time modeling process* of unpredictable processes. We will discuss parallels between unpredictable processes, run time modeling, games, and the proposed approach. Next we will give a case example of how this process could function. This case example will be that of the unpredictable process of the patient in the hospital and several physicians working together and communicating to model the next steps in the treatment plan of the patient. This paper reports the first step in a line of research taking the perspective that we can learn from games to support the management of unpredictable processes. Our goal is to design and create a procedure or method using elements from games to enhance and support the way of working of knowledge workers. We therefore work under the Design Science paradigm [6].

2 Adaptive Case Management: Two Level Approach

Business processes are present within every organization. Managing these business processes is of importance for an organization. 'Business Process Management' (BPM) was introduced to help manage these processes within organizations. BPM traditionally was used to manage predefined workflows [7], but not all processes are routine and well structured. Some processes are unpredictable and knowledge oriented. Adaptive Case Management (ACM) can be used to manage such processes [8,9]. A definition of ACM is given in [10]: "Adaptive Case Management is a collaborative, dynamic, and information-intensive process that is driven by outside events and requires incremental and progressive responses from the business domain handling the case". ACM is designed with the goal to support knowledge workers and their processes in an organization [8,11]. Each instance of a case is unique [11,12], and the process around it is also unique. This paradigm differs from traditional workflow management [8]. Whereas the process in workflow management is always the same and uses a procedural style, the processes in ACM evolve around a specific case (or instance) [8,13]. ACM allows for more flexibility in processes to support variations in the case. Variations in a case occur when an event happens [8,14,15]. These events can be seen as 'dynamic events'. Dynamic events can change the context of a particular

case [14] and can be internal or external [14,16]. To support dynamic events, adaptivity during run time is required [13]. Within ACM we can distinguish two levels of models:

- Fixed Model
- Run time planning

The first is the model in design time based on the meta model for case management models. Such a model consists of different *states* where a case passes through and *plan fragments* from which a caseworker may choose during run time. These plan fragments consist of one or more tasks following some kind of procedure. When dynamic events or changes in a case occur, the system or knowledgeworker has to choose between plan fragments, both available and applicable, to handle this specific instance based on the new context [8,11] and in fact model the actual planning in run time. To model this *run time planning* or 'treatment plan', different roles are often required [17] and should collaborate [11,17,18] to achieve the desired outcome for a case. This paper concerns the modeling of the run time planning of ACM and we propose the idea of using *Gamification* to support this process.

3 Gamification and the Connection to ACM

The use of games within organizations is becoming more mainstream. The term 'Serious Games' is often used, especially in view of management games [3]. Here games are used to learn something about eg. a new method. By using a game format, learning became more enjoyable. The idea this arises of using game elements to make every day work more fun, interesting and user friendly. *Gamification*, a relatively new term which is getting more and more attention. Deterding et al. [5] describe Gamification as: "an informal umbrella term for the use of video game elements in non-gaming systems to improve user experience and user engagement". Next to this improvement, Gamification also aids user friendly conceptualization, communication, visualization and the manipulation of conceptual objects [2,3,5]. As stated in the previous section, during the execution of an instance of a case it is important for all roles to collaborate to commit on a decision. This decision is the next step in the handling of a specific case. By introducing game elements, this collaboration could be supported. McGonigal [19] wrote about collaboration within games: "Gamers agree to play by the same rules and to value the same goal. They practice shared concentration and synchronized engagement". The same is true for the modeling of the run time planning. All roles need to agree to work by the same rules (eg. laws, policies, business rules) and make decisions to achieve a goal (the description and change of the run time planning). This modeling of the run time planning can be seen as a *role playing game* (RPG). We see these RPGs more and more taking form in *Massively Multiplayer Online Games* (MMOG). The biggest missions in MMOGs are called 'Raids'. where players need to work together to defeat

a 'boss'. Raiding represents the most complex form of simultaneous interaction between groups of players and the design structure of the game [20]. Raiding is not just doing a mission together, but is highly collaborative and communicative. Williams and Kirschner [20] stated: "'Raiding' is generally considered among gamers and scholars alike to be the most challenging form of collaborative play". The process of modeling the run time planning is such a collaborative play and much like a raid. Both ACM and games are based on 'meaningful play'. Van Bree and De Lat [21] stated: "In well-designed games we see autonomous individuals, devising short and longer term strategies, reacting to changing situations, absorbing and processing the information needed to complete their task at a fast rate. This behavior is what happens when meaningful play occurs." Salen and Zimmerman [22] described meaningful play as the goal of successful game design and it emerges "from the relationship between player action and system outcome.". To understand game design and to explain how games are quite similar to ACM it helps to distinguish the core elements of games. There are three elements of games described as depicted in Figure 1. The inner circle are the rules of the game, in ACM these can be seen as the business rules and the fixed model. Van Bree and De Lat [21]: "The rule set is communicated through the representation or declarative layer which is shown on the screen". In ACM this is the current state and applicable data of a specific instance of a case. The outer circle depicts where the actual behavior of players takes place. This is the communication between different roles to model the run time planning in ACM. In this paper we see the run time planning process (and communication within it) as the actual work, with the goal to describe and change the run time planning to meet the *objectives* set to complete a case. By designing this process as a game we can inspire effort, reward hard work and facilitate cooperation and collaboration [19] to ultimately enhance the user experience, user engagement, user friendly conceptualization & communication of this process.

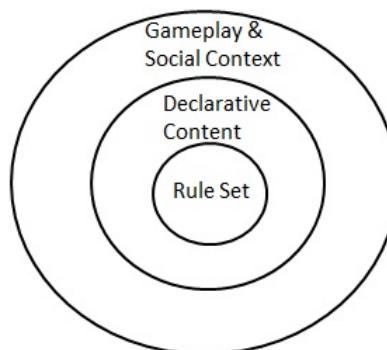


Fig. 1. Elements of games (Salen and Zimmerman [22] & Van Mastrigt [23])

4 Situation and Analysis

The actual modeling process of the run time planning in ACM has a goal driven perspective. Every model serves or works towards at least one clear, utilitarian purpose [24]. In the case of run time planning, it is the description and change of the run time planning. As we stated before, the communication during this process is key. During this collaborative process, different roles "move through a process in which they combine their expertise, insights and their resources to bring them to bear for the task at hand" [25]. If the complex and dynamic collaborative interactions involved are not properly organized and supported, the benefits that potentially accrue from them may not be realized [2]. Ssebuggwawo et al. [2] hypothesize that the interactions that take place in collaborative modeling sessions can be looked at as a game. This section is based on, what in Argumentation Theory [26] is referred to as, Dialogue Games [27]. McGonigal [19] states that a good MMOG has a good game world (eg. players, locations), good game mechanics (eg. game rules, direct & clear results, objectives) and it has a good game community (eg. positive social interaction, meaningful context for collective effort). In the case of the run time planning process in ACM we have the following game elements of an MMOG:

- **Game world:** The *players* in the game are the caseworkers, or the physicians in the health care example. Players can play different kind of *roles*. In ACM there are different kind of knowledge workers which can be involved with the case. In the health care example, we have different kinds of specialists. The *game arena*, or location, is the case itself. In the context of the case, the caseworkers propose their ideas of next steps. Within this game world we can also identify several *game pieces*. In the run time planning process, the game pieces can be identified as the planning fragments in the case.
- **Game mechanics:** By using *game rules*, communication during the run time planning process can be structured. For this we propose the use of *communication items*, which will be explained later in this section. It is also important to have some direct & clear *goals*. These goals need to be made clear every time this run time planning process starts. Two questions need to be asked. What do we have? & What do we want? To answer the question of 'What do we have?' we need the current case (and all the applicable and available data relevant to the current state of the case) and our plan so far (what steps did we take). And to answer the question of 'What do we want?', we need to know what we want to achieve, what the scope is (the focus) and what indications there are. After we have set the goal of the run time planning process, we can create the *objectives* for this process (the final goal and maybe some sub-goals).
- **Game community:** The social interaction between the experts can be supported by the game mechanics stated before. This interaction must always be in context of a *collective effort*. Next to the goal of this run time planning process, this is (in context of the health care example) to make the patient well enough to leave the hospital.

Conversation during processes like the run time planning process involves negotiation, which results in accepts, rejects, modifications , etc., [2] (see, for example, [27]). To support this negotiation during the run time planning process, we propose the use of several *communication items* based on the 'Speech Act Types' proposed by Ssebuggwabo et al. [2] and communication activities proposed by Rittgen [28]. These communication items are listed in Table 1 and can be seen as a game mechanic.

Table 1. Communication items

Communication item	Purpose
(Counter)Propose	Proposing or counter proposing a planning fragment
Argue For	Providing an argument for the proposed planning fragment
Argue Against	Providing an argument against the proposed planning fragment
Agree with/Commit	Agree on or commit to the proposed planning fragment
Disagree with/Reject	Disagree on or reject the proposed planning fragment
Ask Question	Asking a question about the proposed planning fragment
Pass	No contribution at this moment

The communication items can be seen as the *moves* a player can make during this run time planning process. It is also possible for a player to pass when the player has nothing to add at this moment. By using Gamification we can organize the interactions during this process, so benefits that potentially accrue from them can be realized. The suggested game elements, how they map on ACM terms and their link to the health care example (which is explained further in the next section) are shown in Table 2 .

Table 2. Link between game elements, ACM terms & Health care example

Game elements	ACM terms	Health Care Example
Players	Caseworkers	Physicians
Game arena	Case/Run time planning	Treatment plan
Game pieces	Plan fragments	Proposals (eg. tests)
Game play (turn-based)	Modeling process	Creating a treatment plan
Moves (communication items)	Communication on the run time planning	Communication on the treatment plan
Game rules	Design time model	Eg. Policies, Laws, Business Rules & Who makes final accept/reject
Roles	Knowledge workers	Specialists
Objectives	Goals	Goals

By structuring the team play in this modeling process, by introducing these game elements, we hope the entire process could become more user friendly, dynamic, flexible, purposeful, efficient & effective.

5 Example of Team Play Design

To show how these Gamification additions to the run time planning process in ACM might function, we will now give an example. As stated before, the example of a case we will give is that of the treatment of a patient in a hospital. This is a very knowledge intensive and unpredictable process. At a high level we can make a design of this process from the viewpoint of the physician. Eg. Admission, diagnose, treat & dismiss. These can be seen as the states a case can be in. While in a state, the case worker (in this case the attending physician) can choose how to act. Especially in the 'diagnose' & 'treat' states, the process can be highly unpredictable. Now we will give an example of how such a case might be managed and specifically how the run time planning process might take place.

A patient is admitted to the hospital, where a physician is assigned to the patient. The first step is to diagnose the patient. The modeling of the run time planning starts here. This modeling process can be done by the physician her/himself. The physician changes the run time planning based on the applicable and available data in the case (eg. anamnesis and current symptoms), and maybe consolidates other physicians. The next time this modeling process takes place is when a dynamic event happens. This can be when eg. a result from an ordered test is received. Based on this new data (and change in the case), the modeling process of the run time planning starts. In Table 3 we listed a simplistic example of a conversation being done by the physicians proposing a test regarding the case. During this conversation we can see the use of several communication items. In Table 4 we can see which communication items were used. The conversation during the process of modeling the run time planning starts off by introducing a proposal. This proposal, a planning fragment in ACM, is the game piece a player (physician) places on the game arena (the model of the run time planning of the case). This process could eventually be visualized by placing a physical game piece in some sort of model of the game arena.

Table 3. Example conversation during modeling of run time planning

Player	Conversation
Physician A	Because of " <i>some indication</i> " I would like to propose we do a blood test
Physician B	I agree to this proposal, because a blood test could rule out " <i>some disease</i> "
Physician C	I do not think that we should do a blood test, because " <i>some argument</i> "
Physician A	...
Physician B	I do not agree with you, because of " <i>some argument</i> " I think we should do a blood test first
Physician C	...
Physician A	That is indeed why I suggested the blood test
Physician B	Agreed
Physician C	You are correct, lets first do a blood test
Physician A	Okay so our next step is to do this blood test

Table 4. Communication items in conversation

Conversation	Communication item
A: ... I would like to propose ...	Propose
B: I agree ..., because ...	Argue For
C: I do not think that we should do ... because ...	Argue Against
A: ...	Pass
B: I do not agree with you, because ... we should do ...	Argue For
C: ...	Pass
A: That is ... why I suggested ...	Argue For
B: Agreed	Agree with/Commit
C: You are correct ...	Agree with/Commit
A: Okay so our next step is ...	Agree with/Commit

The players each take a turn to make a move, using a communication item. In our example, at the end of the conversation all the players agree/commit to the proposal. It could also be the case that consensus is not reached between players. To make a final decision we should look at the game rules of the process. They might state that there will be a senior player (eg. the attending physician) that will make the final decision whether or not the proposal is accepted or rejected. Or the game rules might state than a certain percentage should agree to the proposal before it is accepted (this might vary from 50% to 100%). When a proposal is accepted or rejected there may be another proposal by a player. When more than one proposal has been accepted, the same process can also start for the temporal ordering between the next steps. By the use of Gamification we can organize this kind of communication about a case.

6 Conclusion and Future Research

As knowledge oriented processes become more mainstream within organizations, and the need to manage these (mostly) unpredictable processes with ACM, we have argued in favor of the use of Gamification to support the process of modeling the run time planning. We have discussed recent work on ACM and discussed the two levels of this approach, where we focused on the planning of next steps at run time. We also presented why we think we can learn from games and how Gamification could support this planning process, and specifically the communication during the modeling of this run time planning. We concluded that we could organize this process by using several game elements. Such as the use of: *players, game arena, game pieces, game play, moves in the game, game rules, roles & objectives*. We have only described one piece of ACM which can be supported by Gamification. There are still some other areas left in ACM where Gamification could provide some support, such as the execution of the steps in run time & the (collaborative) design of a case in the design time of ACM. To help establish these research streams we have argued how the modeling of the run time planning can be supported by Gamification. This theory contributes to

Gamification and ACM research by providing conceptual constructs about Gamification, ACM and a basis for enhancing a process with the use of Gamification. Furthermore, to fully understand how Gamification can be used to enhance the modeling process of the run time planning, an initial pilot game could be created and tested in a knowledge intensive organization (eg. a hospital). In the near future, we plan to carry on in this line of work in a recently started PhD project that this paper is a first product of. Our applied aim is to lay a foundation for Gamification to support ACM.

References

1. Veldhuijen van Zanten, G., et al.: System Development as a Rational Communicative Process. *Journal of Systemics, Cybernetics and Informatics* 2(4), 47–51 (2004)
2. Ssebuggwawo, D., Hoppenbrouwers, S., Proper, E.: Interactions, Goals and Rules in a Collaborative Modelling Session. In: Persson, A., Stirna, J. (eds.) PoEM 2009. LNBIp, vol. 39, pp. 54–68. Springer, Heidelberg (2009)
3. Hoppenbrouwers, S., et al.: Method Engineering as Game Design: an Emerging HCI Perspective on Methods and CASE Tools (2012)
4. Hoppenbrouwers, S.J.B.A., Proper, H.A(E.), van der Weide, T.P.: A Fundamental View on the Process of Conceptual Modeling. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 128–143. Springer, Heidelberg (2005)
5. Deterding, S., et al.: Gamification. using game-design elements in non-gaming contexts. In: CHI EA 2011 CHI 2011 Extended Abstracts on Human (2011)
6. Hevner, A.R., et al.: Design science in information systems research. *MIS Quarterly* 28(1), 75–106 (2004)
7. van der Aalst, W., van Hee, K.: Workflow Management: Modelen, Methoden en Systemen, 2nd edn. Academic Service (2004)
8. Swenson, K., et al.: Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done. MKPress (2010)
9. Swenson, K., et al.: Taming the Unpredictable Real World Adaptive Case Management: Case Studies and Practical Guidance. Future Strategies Inc. (2011)
10. Moore, C., Le Clair, C.: Dynamic Case Management: An Old Idea Catches New Fire. Forrester Research (2009)
11. Weeks, D.E.: Adaptive Case Management: Taming Unstructured Process Work for Today's Knowledge Worker. OpenText Global 360 (2011)
12. Chevron: Future of BPM survey, Chevron (2011)
13. Silver, B.: Case Management Solutions, Case Management Demystified. Webinar Global 360, BPMS Watch (2009)
14. Mecella, M.: Adaptive Process Management. Issues and (Some) Solutions. In: Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2008, pp. 227–228 (2008)
15. Kemsley, S.: Enterprise 2.0 Meets Business Process Management. In: Handbook on Business Process Management 1 International Handbooks on Information Systems, pp. 565–574 (2010)
16. Antic, D., et al.: On a New Approach To The Business Processes Modeling. Automatic Control and Robotics 10(2), 199–204 (2011)

17. Miller, D., Shepherd, T.: Winning in the New Normal: Adaptive Case Management Strategies to Deal with Business as it Happens. Whitepaper Global 360 (2011)
18. Moore, C.: Delivering exceptional customer service. Webinar global 360, Forrester Research (2011)
19. McGonigal, J.: Reality is broken: Why games make us better and how they can change the world. Producer, Massachusetts (2011)
20. Williams, J.P., Kirschner, D.: Coordinated Action in the Massively Multiplayer Online Game World of Warcraft. *Symbolic Interaction* 35(3), 340–367 (2012) ISSN: 0195-6086 print/1533-8665
21. van Bree, J., de Lat, S.: Complex Systems and Emergent Behavior: Engaging with Computer Games to Enrich Organization Studies. Nyenrode Research Paper no. 11-05 (2011) ISSN 1872-3934
22. Salen, K., Zimmerman, E.: Rules of play: game design fundamentals. MIT Press, Cambridge (2004)
23. van Mastrigt, J.: The big bang. Dutch Gamedays, Utrecht (2006)
24. van Bommel, P., Hoppenbrouwers, S.J.B.A., Proper, H.A., Roelefs, J.: Concepts and Strategies for Quality of Modeling. In: Halpin, T.A., Krogstie, J., Proper, H.A. (eds.) Innovations in Information Systems Modeling, ch. 9. IGI Publishing, Hershey (2008)
25. de Vreede, G.J., Briggs, R.O.: Collaboration Engineering: Designing Repeatable Processes for High-Value Collaborative Task. In: Proceedings of the 38th Hawaii International Conference on Systems Science, p. 17c. IEEE Computer Society Press, Los Alamitos (2005)
26. van Eemeren, F.H., et al.: Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Developments. Lawrence Erlbaum Associates, Mahwah (1996)
27. Hoppenbrouwers, S.J.B.A., Proper, H.A., van der Weide, T.: Formal Modelling as a Grounded Conversation. In: Goldkuhl, M., Lind, G., Haraldson, S. (eds.) Proceedings of the 10th International Working Conference on the Language Action Perspective on Communication Modelling, LAP 2005, Kiruna, Sweden, pp. 139–155. Linkpings Universitet and Hogskolan I Boras, Linkping (2005)
28. Rittgen, P.: Negotiating Models. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 561–573. Springer, Heidelberg (2007)

A Semiotic Approach to Data Quality

John Krogstie

Norwegian University of Science and Technology (NTNU)
krogstie@idi.ntnu.no

Abstract. Since the introduction of the ER-language in the late seventies, data modeling has been an important aspect of information systems development. The quality of data models has been investigated since the mid-nineties. In another strand of research, data and information quality has been investigated even longer. Data can also be looked upon as a type of model (on the instance level), as illustrated e.g. in the product models in CAD-systems. In this paper we present a specialization of a general framework for assessing quality of models to be able to evaluate the combined quality of data models and data. A practical application of the framework from assessing the potential quality of different data sources to be used in a collaborative work environment is used for illustrating the usefulness of the framework. We find on the one hand that the traditional properties of data quality and data model quality is subsumed by the generic SEQUAL-framework, and that there are aspects in this framework that are not covered by the existing work on data and data model quality. On the other hand, the comparison has resulted in a useful deepening of the generic framework for data quality, and has in this way improved the practical applicability of the SEQUAL-framework when applied to discussing and assessing data quality.

1 Introduction

Data quality has for a long time been an established area discussing wanted properties of data [1]. A related area that was established in the nineties is quality of models (in particular quality of conceptual data models) [16]. Traditionally, one has here looked at model quality for models on the M1 (type) level (to use the levels found in e.g. MOF [3]). On the other hand, it is clear especially in product and enterprise modeling that there are models on the M0 or instance level, an area described as containing data (or objects in MOF-terminology). Thus our hypothesis is that also data quality can be looked upon relative to more generic frameworks for quality of models. Comprehensive and generic frameworks for evaluating modelling approaches has been developed [10, 14, 22], but these can become too general for practical use.

Discussions on data quality must be looked upon in concert with discussions on data model (or schema) quality. Inspired by [19], suggesting the need for an inheritance hierarchy of quality frameworks, we will in this paper provide a specialization of the generic SEQUAL framework [10] for the evaluation of the quality of data and their accompanying data models. Similar specializations of SEQUAL for the quality

of conceptual data models [12], business process models [11] and software requirements specifications [7] have earlier been reported.

In section 2, we present selected existing work on data and data model quality. Section 3 provides a brief overview of SEQUAL, whereas the specialization of SEQUAL for data quality is described in section 4. The specialization adopts a conceptual-analytical approach, providing an informed argument building on the existing knowledge base. An example of action research, using the framework in practice is provided in section 5. The main research questions for the work, inspired by results from earlier specializations [7, 10, 11, 12], are:

- Is it possible to specialize the SEQUAL framework covering the aspects of data quality described in the literature?
- Will this specialization introduce new areas of concern for data quality?
- Is this specialization found useful for practical evaluation of data quality

In section 6, we sum up planned work for developing and evaluating an integrated approach for data and data model quality.

2 Existing Work on Data Quality and Quality of Data Models

There are a number of approaches to defining dimensions of data quality. We will base this section on the framework presented in [1], where the following aspects are discussed relative to the data values in a relational database.

When looking upon data quality in isolation, the underlying data model can be looked upon as part of the context (i.e. a pre-existing model that this model should

Table 1. Dimensions of data quality

Dimension name	Sub-Category	Definition
Accuracy	Syntactic	Distance between v (the correct value) and v' (the incorrect value)
	Semantic	
Completeness		Degree to which all values are present in a data collection
Time-related aspects	Currency	Degree to which the data is up-to-date
	Volatility	Frequency with which data vary with time
	Timeliness	How current the data is for the task at hand
Consistency		Coherence of the same datum, represented in multiple copies, or different data to respect integrity constraints and rules
Interpretability		Concerns the documentation including the data model, and other metadata that are available to correctly interpret the meaning of data
Accessibility		Data is accessible for those needing access to the data in a format that can be understood
Quality of information source	Believability	Is the data provided true, real and credible?
	Reputation	Is the source normally credible?
	Objectivity	Is the source believed to be objective?

relate to). Obviously how the data is meant to be and actually are used influences the perceived quality of the data. To also capture this aspect, Price et al. [23, 24] use the term *information quality* to combine a product-based and service-based view. The product-based perspective, covered by traditional data quality properties as described above, focuses on the internal IS view. From this view, quality is defined in terms of the degree to which the data meets initial requirements specifications or the degree to which the data corresponds to the relevant real-world phenomena that it represents. The limitation with this is that even if data corresponds to a requirements specification or the real-world, there can still be quality deficiencies with respect to actual use-related data requirements, which may differ from the planned uses catered for in the initial specifications. This leads to a service-based perspective of data quality, called *information quality*, which focuses on the information consumer's response to their task-based interactions with the IS. Price and Shanks defines this area in the following way, building upon semiotic theory [21]. Based on empirical evaluations of the original framework presented in 2004 [23], the following quality categories have been defined [24]:

Syntactic Criteria (based on rule conformance)

- *Conforming to metadata, i.e. integrity rules.* Data follows specified database integrity rules.

Semantic Criteria (based on external correspondence)

- *Mapped completely.* Every real-world phenomenon is represented.
- *Mapped unambiguously.* Each identifiable data unit represents at most one specific real-world phenomenon.
- *Phenomena mapped correctly.* Each identifiable data unit maps to the correct real-world phenomenon.
- *Properties mapped correctly.* Non-identifying (i.e. non-key) attribute values in an identifiable data unit match the property values for the represented real-world phenomenon.
- *Mapped consistently.* Each real-world phenomenon is either represented by at most one identifiable data unit or by multiple, but consistent identifiable units or by multiple identifiable units whose inconsistencies are resolved within an acceptable time frame.
- *Mapped meaningfully.* Each identifiable data unit represents at least one specific real-world phenomenon.

Pragmatic Criteria (use-based consumer perspective)

- *Accessible (easy, quick).* Data is easy and quick to retrieve.
- *Suitably presented (suitably formatted, precise, and measured in units).* Data is presented in a manner appropriate for its use, with respect to format, precision, and units.
- *Flexibly presented (easily aggregated; format, precision, and units easily converted).* Data can be easily manipulated and the presentation customised, with respect to aggregating data and changing the data format, precision, or units.
- *Timely.* The currency (age) of the data is appropriate to its use.
- *Understandable.* Data is presented in an intelligible manner.
- *Secure.* Data is appropriately protected from damage or abuse (including unauthorised access, use, or distribution).

- *Type-sufficient.* The data includes all of the types of information important for its use.
- *Allowing access to relevant metadata.* Appropriate metadata is available to define, constrain, and document data.
- *Perceptions of the syntactic and semantic criteria defined earlier.*

As discussed in [1] data quality should be looked upon in connection to quality of the underlying data models. Data models are a type of structural models used both for human sense-making and communication and as a context for systems development. Approaches within the structural modelling perspective concentrate on describing the static structure of a domain. Going back to the ANSI SPARC work [28], one differentiates between three levels of data models: Conceptual models (e.g. ER models), logical models (e.g. in the form of relational tables), and physical models (e.g. a physical implementation of a relational database using a DBMS).

There exist well-defined mappings between these levels, although often automatic mappings are not sufficient in practice to get ideal database performance based on the conceptual and logical models.

Some of the early work on quality of models focused on data models [16], a quality model that was extended in [17, 18] based on empirical investigations on its use.

The quality model in [18] contains the following properties:

- *Correctness* is defined as whether the model conforms to the rules of the data modelling technique. This includes diagramming conventions, naming rules, definition rules, rules of composition and normalisation.
- *Completeness* refers to whether the data model contains all information required to support the required functionality of the system.
- *Integrity* is defined as whether the data model defines all business rules that apply to the data.
- *Flexibility* is defined as the ease with which the data model can cope with business and/or regulatory change.
- *Understandability* is defined as the ease with which the concepts and structures in the data model can be understood.
- *Simplicity* means that the data model contains the minimum possible entities and relationships.
- *Integration* is defined as the consistency of the data model with the rest of the organisation's data.
- *Implementability* is defined as the ease with which the data model can be implemented within the time, budget and technology constraints of the project.

Another overview of data model (schema) quality is presented in [1], including:

- *Correctness* with respect to the model concerns the correct use of the concepts in the language. A negative example is to represent FirstName as an entity, and not as an attribute (since FirstName can be argued to not have unique existence)
- *Correctness* with respect to requirements
- *Minimalisation*, no requirement is represented more than once
- *Completeness*
- *Pertinence* that measures the number of unnecessary conceptual elements

- *Readability* through aesthetics
- *Readability* through simplicity
- *Normalisation*

3 Introduction to SEQUAL

SEQUAL [10] is a framework for assessing and understanding the quality of models and modelling languages. It has earlier been used for evaluation of modelling and modelling languages of a large number of perspectives, including data [12], object [8], process [11, 25], enterprise [13], and goal-oriented [6, 9] modelling. Quality has been defined referring to the correspondence between statements belonging to the following sets:

- G , the set of goals of the modelling task.
- L , the language extension.
- D , the domain, i.e., the set of all statements that can be stated about the situation. Domains can be divided into two parts, exemplified by looking at a software requirements specification:
 - Everything the computerized information system (CIS) is supposed to do. This is termed the *primary domain*.
 - Constraints on the model because of earlier baselined models such as system level requirements specifications, enterprise architecture models, statements of work, and earlier versions of the requirement specification. This is termed the *modelling context*. In relation to data quality, the underlying data model is part of the modelling context.
- M , the externalized model itself.
- K , the explicit knowledge relevant to the domain of the audience.
- I , the social actor interpretation of the model
- T , the technical actor interpretation of the model

The main quality types are:

- Physical quality: The basic quality goal is that the externalized model M is available to the relevant social and technical actors (and not to others).
- Empirical quality deals with comprehension and predictable error frequencies when a model M is read by different social actors
- Syntactic quality is the correspondence between the model M and the language extension L .
- Semantic quality is the correspondence between the model M and the domain D .
- Perceived semantic quality is the similar correspondence between the social actor interpretation I of a model M and his or hers current knowledge K of domain D .
- Pragmatic quality is the correspondence between the model M and the actor interpretation (I and T) and application of it.
- The goal defined for social quality is agreement among actor's interpretations.
- The deontic quality of the model relates to that all statements in the model M contribute to fulfilling the goals of modelling G , and that all the goals of modelling G are addressed through the model M .

4 Data Quality Relative to the SEQUAL Quality Types

We here discuss means within each quality level, positioning the areas that are specified by *Batini* et al. [1] and *Price* et al. [23, 24]. These are emphasised using italic.

4.1 Physical Data Quality

Aspects of persistence, data being *accessible* (*Price*) for all (*accessibility* (*Batini*)), *currency* (*Batini*) and *security* (*Price*) cover aspects on the physical level. This area can be looked upon relative to measures of persistence, currency and availability that apply also to all other types of models. Tool functionality in connection with physical quality is based on traditional database-functionality.

4.2 Empirical Data Quality

This is addressed by *understandable* (*Price*). Since data can be presented in many different ways, this relates to how the data is presented and visualized. How to best present different data depends on the underlying data-type. There are a number of generic guidelines within data visualization and related areas that can be applied, and we will only mention a few of these here. For computer-output specifically, many of the principles and tools used for improving human computer interfaces are relevant at the empirical level [27]. For visual presentation of data, one can also base the guidelines on work in cognitive psychology and cartography with the basis that data is meant to be useful in connection to *communication* between people. Going back to [26], communication entails both encoding by the sender and decoding by the receiver. Encoding has been discussed in detail e.g. in the work of Bertin [2]. According to [2] there are 4 different effects of encoding:

1. Association. The marks can be perceived as similar
2. Selection. The marks can be perceived as different
3. Order. The marks can be perceived as ordered
4. Quantity. The marks can be perceived as proportional

8 different variables presented to convey one or more of these meanings in a visualisation are:

- planar variables: horizontal position, vertical position
- retinal variables: shape (association and selection), size (selection, order and quantity), colour (Association and selection), brightness (value) (selection and order), orientation (association), texture (association, selection and order).

For decoding Moody [20] presents a model differentiating between aspects of perception and cognition.

- Perceptual discrimination: Features are detected by specialised feature detectors. Based on this, the visualization is parsed into its parts.
- Perceptual configuration: Structure and relationship among elements are inferred. Within the area of Gestalt psychology, a number of principles for how to convey meaning through perceptual means are provided [29].

- Attention management: All or part of the perceived image is brought into working memory. Working memory has very limited capacity. To be understood, statements in the model must be integrated with prior knowledge in the long-term memory of the interpreter. Differences in prior knowledge (expert-novice differences) greatly affect the speed and accuracy of processing.

Rules for colour-usage are also useful in connection to evaluating data visualization (if different colours are used). Shneiderman [27] has listed a number of guidelines for the usage of colour in visual displays in general. The use of emphasis can also be in accordance with the relative importance of the data. Factors that have an important impact on visual emphasis are:

- Size (the big is more easily noticed than the small)
- Solidity (e.g. **bold** letters vs. ordinary letters, full lines vs. dotted lines, thick lines vs. thin lines, filled boxes vs. non-filled boxes)
- Difference from ordinary pattern (e.g. *slanted* letters will attract attention among a large number of ordinary ones)
- Foreground/background differences (if the background is white, things will be easier noticed the darker they are)
- Change (blinking or moving symbols attract attention)
- Position (looking at a two-dimensional arrangement, people tend to start at its middle)
- Connectivity (objects that connect with many others (having a high degree) will attract attention compared to objects with few connections).

4.3 Syntactic Data Quality

From the generic SEQUAL framework we have that there is one main syntactic quality characteristic, **syntactical correctness**, meaning that all statements in the model are according to the syntax and vocabulary of the language

Syntax errors are of two kinds:

- **Syntactic invalidity**, in which words or graphemes not part of the language are used.
- **Syntactic incompleteness**, in which one lack constructs or information to obey the language's grammar

Conforming to metadata (Price) including that the data conform to the expected data type of the data (as described in the data model) are part of syntactic data quality. This will typically be related to syntactic invalidity when e.g. the data is of the wrong data-type.

4.4 Semantic Data Quality

When looking upon semantic quality relative to the primary domain of modelling, we have the following properties:

Completeness is covered by *completeness* (Batini), *mapped completely* (Price), and *mapped unambiguously* (Price).

Validity is covered by *accuracy* (*Batini*), both syntactic and semantic accuracy as they have defined it, the difference between these is rather to decide on how incorrect the data is, *phenomena mapped correctly* (*Price*), *properties mapped correctly* (*Price*) and *properties mapped meaningfully* (*Price*). Since the rules of representation are formally given, *consistency* (*Batini*)/*mapped consistently* (*Price*) is also related to validity. The use of meta-data such as the source of the data is an important mean to support validity of the data.

Properties related to the model context are related to the adherence of the data to the data model. One would expect for instance that

- All tables of the data model should include tuples
- Data is according to the constraints defined in the data-model

The possibility of ensuring high semantic quality of the data is closely related to the semantic quality of the underlying data model. When looking upon semantic quality of the data model relative to the primary domain of modelling, we have the following properties: *Completeness* (*Moody and Batini*) (number of missing requirements) and *integrity* (*Moody*) (number of missing business rules) relates to completeness.

Completeness (*Moody*) (number of superfluous requirements) and *integrity* (*Moody*) (number of incorrect business rules) relates to validity. The same applies to *Batini's* points on *correctness with respect to model* and *correctness with respect to requirements*.

4.5 Pragmatic Data Quality

Pragmatic quality relates to the comprehension of the model by participants. Two aspects can be distinguished:

- That the interpretation by human stakeholders of the data is correct relative to what is meant to be expressed. In addition to the data it will often be useful to have different meta-data represented (Making it easier to understand the intent behind the data).
- That the tool interpretation is correct relative to what is meant to be expressed.

Starting with the human comprehension part, pragmatic quality on this level is the correspondence between the data and the audience's interpretation of it. Moreover, it is not only important that the data has been understood, but also *who* has understood (the relevant parts of) the data.

The main aspect at this level is *interpretability* (*Batini*), that data is *suitably presented* (*Price*) and data being *flexibly presented* (*Price*). Allowing *access to relevant metadata* (*Price*) is an important mean to achieve comprehension.

4.6 Social Data Quality

The goal defined for social quality is *agreement*. Relative agreement means that the various sets to be compared are consistent -- hence, there may be many statements in the data representation of one actor that are not present in that of another, as long as they do not contradict each other. The area *quality of information source* (*Batini*)

touches important mean for the social quality of the data, since a high quality source this will increase the probability of agreement.

In some cases one need to combine different data sources. This consists of combining the data-models, and then transferring the data from the two sources into the new schema. Techniques for schema integration [4] are specifically relevant for this area.

4.7 Deontic Data Quality

A number of aspects are on this level relating to the goals of having the data in the first place. Aspects do decide *volatility* (*Batini*) and *timeliness* (*Batini*)/*timely* (*Price*) needs to relate to the goal of having and distributing the data. The same is the case for *type-sufficient* (*Price*), the inclusion of all the types of information important for its use.

5 Application of the Framework

LinkedDesign is an ongoing international project that aims to boost the productivity of today's engineers by providing an integrated, holistic view on data, persons and processes across the full product lifecycle. To achieve this there is a need to evaluate the appropriateness of a selected number of existing knowledge sources, to be used as a basis for the support of collaborative engineering in a Virtual Obeya, a kind of collaborative work environment [30]. The selected knowledge sources are of the types found particularly relevant in the use cases of the project. When we look of *quality* of an information source (e.g. a CAD tool), we look on both the structure of the stored data (the data model, including meta-data) and the characteristics of the data itself, in light of our goal for reuse and revisualization of data, in a way that might be annotated and/or updated through the user interface. The users perform collaborative work using the Virtual Obeya. The Obeya presents context specific information based on the persons involved in the collaboration and other relevant information on products, projects, locations, tasks, tools, rules and guidelines etc. The data is mediated from existing work tools and the data have to be transformed depending on the relevant context. The data presented and worked on in the Virtual Obeya can be annotated with other context-oriented information that potentially is stored.

Looking at the sets of SEQUAL in the light of this case, we have the following:

- G: There are goals on two levels. The goal to be achieved when using the base tool, and the goal of supporting collaborative work using data from this tool as one of several sources of knowledge for the Virtual Obeya. Our focus is on this second goal, although these might be related.
- L: The language is the way data is encoded (e.g. using some standard), and the language for describing the data model/meta-model.
- M: Again on two levels, the data and the data-model.
- A: Actors i.e. the people in different roles using the models, with a specific focus on the collaborators in the use-cases of the project.
- K: The relevant explicit knowledge of the actors (A) in these roles
- T: Relates to the possibilities of the languages used to provide tool-support in handling the data (in the base tools, and also in a Virtual Obeya)

- I: Relates to how easy it is for the different actors to interpret the data as it can be presented (in the base tool, and also in a Virtual Obeya)
- D: Domain: The domain can on a general level be looked upon relative to the concepts of an upper-level ontology. We focus on perspectives captured in the generic EKA - Enterprise Knowledge Architecture of Active Knowledge Models (AKM) since these have shown to be useful for context-based user interface development in other projects [14]. Thus we look on information on: Products, tasks, goals and rules (from standards to design rules), roles (including organizational structure and persons, and their capabilities) and tools.

Based on this we can describe the quality of data more precisely in the following way:

- **Physical quality** relates to if the data is:
 - Available in a physical format in a timely manner so that it can be reused in the Virtual Obeya. The availability of data to be used in other types of tools is also relevant here.
 - Availability of different versions of the data when relevant.
 - Possibility to store relevant meta-data (e.g. the user that has made a CAD model, time-stamp to judge currency etc.)
 - Available for update or annotation/extension in the user interface
 - Availability of data from other tools
 - Only available for those that should have access (security)
- **Empirical quality** is not directly relevant when evaluating the data-sources per se. Guidelines for this is relevant when we look upon how data can be presented in tools (and in the Virtual Obeya itself) (e.g. as a visualized CAD-drawing that can be manipulated in a 3D-interface)
- **Syntactic quality**. Is the data represented in a way following the defined syntax?
- **Semantic quality**. Do the data sources potentially contain the expected type of data? This is looked upon relative to the domain identified on the upper level of the ontology/context-model as discussed above, although a single data source will seldom have data of all relevant types. Note that we here look on the possibility of representing the relevant types of data, obviously the level of completeness is dependent on what is represented in the concrete case. Tools might also have mechanisms for supporting the rapid development of complete models.
- **Pragmatic quality**. Is data of such a type that it can be easily understood (or visualized in a way that is can be easily understood) by the stakeholders. Since the context of work is varying when using the Virtual Obeya, it is important that data can be flexibly represented, including access to relevant meta-data. Possibility of tool interpretation can also be important, given that one need to identify the relevant aspects to show in the Virtual Obeya based on the relevant context. Since there is limited focus on process automation in this particular project, the need for representations with an executional semantics is limited.
- **Social quality**. Is there agreement on the quality of the data among the stakeholders? Since different data comes from different tools, and often need to be integrated in the Virtual Obeya, agreement on interpretation of data and of the quality of this data across the involved stakeholders can be important.

- **Deontic quality:** Shall we with the help of data from the data source be able to achieve the goals of the project? An important aspect of the case project relate to reducing waste in lean engineering processes [15].

5.1 Evaluations of Relevant Tool-Types

In this project, based on the needs of the use cases, we have focused on the following concrete tools and tool types in the assessment.

- Office automation: Excel (could also be Sharepoint, Word, Powerpoint etc)
- Computer-Aided Design (CAD): PDMS, Autocad, Catia V5
- Knowledge-based Engineering (KBE): KBEdesign
- Product Lifecycle Management (PLM/ PDM): Teamcenter, Enovia
- Enterprise Research Planning (ERP): SAP ERP (R/3), MS Dynamics

In the following we present the treatment of one of these areas, the quality of data in CAD tools. Similar evaluations have been done on the other tool-types, and we will summarize in the end of this section how this has been useful for the project.

CAD (Computer Aided Design)-tools are tools used to assist in the creation, modification, analysis, or optimization of the design of a product. CAD software uses either vector based graphics to depict the objects of traditional drafting, or may also produce raster graphics showing the overall appearance of designed objects. Whereas CAD traditionally was used by product designers, functionality to support different engineering professions are often included, and the term CAE - Computer Aided Engineering is also often used for this tool type. A large number of CAD-tools exist (e.g. Autocad, Autodesk, PDMS, SolidWorks, Unigraphics, Ideas NX, Solid Edge, Catia V5, Pro Engineer).

5.2 Features Supporting Physical Quality of CAD Data

Data is stored in a local database, and can be (partly) exchanged based on standard representations (see more under syntactic quality). Tools such as PDMS support simultaneous work by multiple users (from multiple disciplines), with support for versioning and access control. In many tools, not only the product-information is stored, but also the history-tree of operations done for producing the model. Exchange with other tools is described below. It is also possible to export data in generic formats (for visualization) like PDF. In these formats you cannot change the product-data directly.

5.3 Features Supporting Empirical Quality of CAD Data

CAD tools typically have good functionality to visualize the product data in 3D. Because of its economic importance, CAD has been a major driving force for research in computational geometry and computer graphics and thus for algorithms for visualizations that one typically focus on as means under the area of empirical quality.

5.4 Features Supporting Syntactic Quality of CAD Data

The international CAD data exchange standard including IGES and ISO 10303. ISO 10313 is informally known as STEP – STandard for Exchange of Product model data,

has so far been limited to transfer of geometry-information. Note that when exporting the model in STEP or IGES, the history-tree is not included. These standards have been incapable of handling parameters, constraints, design features and other ‘design intent’ data generated by modern CAD systems [5]. In the ProSTEP iViP project the new STEP AP 242 standard with improved capabilities in this respect was implemented and tested in a consortium including the vendors of CATIA, Pro/Engineer and Siemens NX, but is not supported yet in the main CAD-systems.

5.5 Features Supporting Semantic Quality of CAD Data

CAD-systems typically focus on the (geometric) representations of products at the instance level. One might also represent some rules relative to the product in CAD systems (relative to e.g. the materials, processes, dimensions, and tolerances involved), but one does not capture knowledge on organizational structure, tools, and underlying business processes in these tools. Another limitation in most CAD-tools is lacking representation of the function (e.g. overall goal) of the different parts of the design, although some tools support representation of (functional and non-functional) requirements together with the product-information.

CAD tools typically support the development of catalogues of elements. The catalogue contains standard reference data for the available types of components. This can support rapid development of new structures, i.e. support rapid achievement of completeness of the product model. The support of default values in the tool user interface can also be useful in this respect. CAD tools are often integrated with different analysis tools (e.g. for finite element analysis) which can support the development of valid design-models.

5.6 Features Supporting Pragmatic Quality of CAD Data

CAD tools support numerous ways of visualizing the design in an integrated manner, e.g. 3D-view, product model trees etc. Viewing mechanisms showing only parts of the overall structure through layering (e.g. the parts relevant for one discipline), relating this to the overall structure is a very important mechanism to be able to handle the complexity of CAD models. Another interesting approach for ensuring comprehension is the export of the CAD-data to prototyping tools such as 3D-printers which recently has become much cheaper. While the goal of automated CAD systems is to increase efficiency, they are not necessarily the best way to allow newcomers to understand the geometrical principles of solid modeling. For this, scripting languages such as PLaSM (Programming Language of Solid Modeling) have been developed. The scripting approach is very different from working with an interactive GUI, but is preferred by some CAD instructors as scripts reveal all details of the design procedure (not only the final design).

5.7 Features Supporting Social Quality of CAD Data

CAD is primarily used by the designers and some of the engineers, often in an early stage of product development. Thus the organizational agreement on these data might be less than e.g. data developed in more organizationally integrating tools such as

PLM and ERP-tools. On the other hand, the part of the data that is stored using established standards (e.g. STEP) would probably be easier to ensure that is interpreted identically across user-groups and also across organizations, with the limitation that not all the relevant information is captured in these interchange formats. STEP is developed and maintained by the ISO technical committee TC 184, *Automation systems and integration*, sub-committee SC 4, *Industrial data*. Like other ISO and IEC standards STEP is copyright by ISO and is not freely available. However, the 10303 EXPRESS schemas are freely available, as are the recommended practices for implementers. Also since important data relates to representation of physical products it is probably easier to agree on than more conceptual data (e.g. through the development of physical prototypes).

5.8 Features Supporting Deontic Quality of CAD Data

Looking upon the main waste forms in lean engineering [15] we can say the following:

- Searching: Finding the relevant CAD data can be made easier by linking it to enterprise tools such as PLM tools.
- Under-communication: In CAD tools, there might be limitations in the representation of underlying design rules and process information (e.g. relevant for manufacturing) that can be useful at a later stage.
- Misunderstanding: Due to the number of assumptions that are included in a product design that is not necessarily kept explicit, there might be misunderstandings at a later stage.
- Interpreting: A number of tools exist to do different types of analysis, which might make it easier to support interpretation of the product model.
- Waiting: If other than designers and engineers need information from CAD tools, or need to have changes done at a later stage, they might be dependent on the availability of the engineer to do the changes.
- Extra processing: Since CAD tools store the geometry on the instance level; reuse for e.g. variants of products might take extra time.

5.9 Summary of Result from Evaluation

Above, we have seen one of the five assessments done using the specialization of SEQUAL for data quality. The overall evaluation has indicating opportunities, but also challenges when trying to integrated data from different knowledge sources typically used by people in different roles in an organization, and presenting this in a common user interface, supporting collaboration between these people. In particular it highlights how different tools have a varying degree of explicit meta-model (data model), and how this is available externally in a varying degree. E.g. in many export-formats one loses some of the important information on product data. Even when different tools support e.g. process data, it is often process data on different granularity which might be difficult to integrate. The tools alone all have challenges relative to waste in lean engineering. In a Virtual Obeya environment one would want to combine data from different sources in a context-driven manner to address these reasons

for waste. A bit dependent on the concrete knowledge sources to combine, this indicates that it is often a partly manual job to prepare for such matching. Also the different level of agreement of data from different sources (social quality) can influence the use of schema and object matching techniques in practice.

6 Concluding Remarks

As with the quality of a software requirements specification (SRS) [7] and BPM [11], we see some benefit both for SEQUAL and for a framework for data quality by performing this kind of exercise, returning to our research questions:

- Existing work on data and information quality, as summarized in [1, 23, 24] can be positioned within the generic SEQUAL framework as described in Section 4.
- These existing overviews are weak on explicitly addressing areas such as empirical and social quality, as also described in Section 4.2 and Section 4.6.
- The work by Batini and Price et al. on the other hand enriches the areas of in particular semantic and pragmatic data quality, as described in section 4.4 and section 4.5.
- The framework, especially the differentiation between the different quality levels has been found useful in the case from which we have partly reported in Section 5, since it highlights potential challenges of matching data from different sources as discussed in Section 5.8.

Future work will be to device more concrete guidelines and metrics and evaluate the adaptation and use of these empirically in other cases, especially how to perform trade-offs between the different quality types. Some generic guidelines for this exist in SEQUAL [10], which can be specialised for data quality and quality of conceptual data models.

Acknowledgements. The research leading to these results was done in the Linked-Design project that has received funding from the European Union Seventh Framework Programme under grant agreement n°284613.

References

1. Batini, C., Scannapieco, M.: *Data Quality: Concepts, Methodologies and Techniques*. Springer (2006)
2. Bertin, J.: *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press (1983)
3. Booch, G., Rumbaugh, J., Jacobson, I.: *The Unified Modeling Language: User Guide*, 2nd edn. Addison-Wesley (2005)
4. Francalanci, C., Pernici, B.: View integration: A survey of current developments. Technical Report 93-053, Politecnico de Milano, Milan, Italy (1993)
5. Kim, J., Pratt, M.J., Iyer, R., Sriram, R.: Data Exchange of Parametric CAD Models Using ISO 10303-108, NISTIR 7433 (2007)

6. Krogstie, J.: Using Quality Function Deployment in Software Requirements Specification. Paper presented at the Fifth International Workshop on Requirements Engineering: Foundations for Software Quality, REFSQ 1999, Heidelberg, Germany, June 14–15 (1999)
7. Krogstie, J.: A Semiotic Approach to Quality in Requirements Specifications. In: Liu, K., Clarke, R.J., Andersen, P.B., Stamper, R.K., Abou-Zeid, E.-S. (eds.) *Organizational Semiotics*. IFIP, vol. 94, pp. 231–249. Springer, Boston (2002)
8. Krogstie, J.: Evaluating UML Using a Generic Quality Framework. In: Favre, L. (ed.) *UML and the Unified Process*, pp. 1–22. IRM Press (2003)
9. Krogstie, J.: Integrated Goal, Data and Process modeling: From TEMPORA to Model-Generated Work-Places. In: Johannesson, P., Søderstrøm, E. (eds.) *Information Systems Engineering From Data Analysis to Process Networks*, pp. 43–65. IGI (2008)
10. Krogstie, J.: Model-based development and evolution of information systems: A quality approach. Springer, London (2012)
11. Krogstie, J.: Quality of Business Process Models. In: Sandkuhl, K., Seigerroth, U., Stirna, J. (eds.) *PoEM 2012. LNBP*, vol. 134, pp. 76–90. Springer, Heidelberg (2012)
12. Krogstie, J.: Quality of Conceptual Data Models. In: Proceedings 14th ICISO, Stockholm, Sweden (2013)
13. Krogstie, J., Arnesen, S.: Assessing Enterprise Modeling Languages using a Generic Quality Framework. In: Krogstie, J., Siau, K., Halpin, T. (eds.) *Information Modeling Methods and Methodologies*. Idea Group Publishing (2004)
14. Lillehagen, F., Krogstie, J.: *Active Knowledge Modeling of Enterprises*. Springer (2008)
15. Manyika, J., Sprague, K., Yee, L.: Using technology to improve workforce collaboration. *What Matters*. McKinsey Digital (October 2009)
16. Moody, D.L., Shanks, G.G.: What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models. In: Loucopoulos, P. (ed.) *ER 1994. LNCS*, vol. 881, pp. 94–111. Springer, Heidelberg (1994)
17. Moody, D.L.: Metrics for Evaluating the Quality of Entity Relationship Models. In: Ling, T.-W., Ram, S., Li Lee, M. (eds.) *ER 1998. LNCS*, vol. 1507, pp. 211–225. Springer, Heidelberg (1998)
18. Moody, D.L., Shanks, G.: Improving the quality of data models: empirical validation of a quality management framework. *Information Systems* 28(6), 619–650 (2003)
19. Moody, D.L.: Theoretical and practical issues in evaluating the quality of conceptual models: Current state and future directions. *Data and Knowledge Engineering* 55, 243–276 (2005)
20. Moody, D.L.: The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering* 35, 756–779 (2009)
21. Morris, C.: Foundations of the Theory of Signs. *International Encyclopedia of Unified Science*, vol. 1. University of Chicago Press, London (1938)
22. Nelson, H.J., Poels, G., Genero, M., Piattini, M.: A conceptual modeling quality framework. *Software Quality Journal* (2011)
23. Price, R., Shanks, G.: A Semiotic Information Quality Framework. In: IFIP WG8.3 International Conference on Decision Support Systems (DSS 2004), Prato, Italy, July 1–3, pp. 658–672 (2004)
24. Price, R., Shanks, G.: A semiotic information quality framework: Development and comparative analysis. *Journal of Information Technology* 20(2), 88–102 (2005)
25. Recker, J., Rosemann, M., Krogstie, J.: Ontology- versus pattern-based evaluation of process modeling language: A comparison. *Communications of the Association for Information Systems* 20, 774–799 (2007)

26. Shannon, C.E., Weaver, W.: *The Mathematical Theory of Communication*. Univ. of Illinois Press (1963)
27. Shneiderman, B.: *Designing the User Interface: Strategies for Effective Human- Computer Interaction*, 2nd edn. Addison Wesley, Reading (1992)
28. Tsichritzis, D., Klug, A.: The ANSI/X3/SPARC DBMS Framework. *Information Systems* 3, 173–191 (1978)
29. Ware, C.: *Information Visualization*. Morgan Kaufmann (2000)
30. Aasland, K., Blankenburg, D.: An analysis of the uses and properties of the Obeya. In: *Proceedings of the 18th International ICE-Conference*, Munich (2012)

Feedback-Enabled MDA-Prototyping Effects on Modeling Knowledge

Gayane Sedrakyan and Monique Snoeck

Katholieke Universiteit Leuven,
Management Information Systems,
Naamsestraat 69, 3000 Leuven

{gayane.sedrakyan, monique.snoeck}@kuleuven.be

Abstract. This paper describes the effects of a feedback-enabled MDA prototyping tool on the validation cycle for conceptual models. We observe the effects of such prototyping method on learning outcomes of novice modelers. The impact is assessed based on the quality dimensions introduced by Conceptual Modeling Quality Framework (CMQF), more specifically with respect to semantic quality being affected by modeling knowledge. The current work proposes an extension to the techniques introduced in previous work in particular, experimenting with the prototyping tool by novice modelers. A positive impact has been observed on the learning achievements of novice modelers improving both modeling and language knowledge.

Keywords: teaching/learning conceptual modeling, model validation, conceptual model quality, model driven architecture/engineering, prototyping, executable model, UML.

1 Introduction

Analysis and modeling of information systems is a cognitive activity the goal of which is to accurately externalize (map) business requirements into a high-quality formal representation, namely a model. The need for formal models is motivated by the fact that formal models enable quality control at a level that is impossible to reach with informal techniques [1]. Key factors involved in this formalization process affecting the quality of a conceptual model are the modeling knowledge, the modeling language knowledge and the modeling domain knowledge among others [2]. Teaching such knowledge and skills is a challenging task. The fact that there is a significant gap between the modeling knowledge and skills of novice modelers and those of expert modelers generates the question of how these skills can be taught to improve the understanding of modeling as well as the effectiveness of new modelers [2]. Currently effective guidance on learning conceptual modeling as well as tool support is largely lacking. Furthermore, there are aspects that cannot be gained with reading or lecturing alone, e.g. the dynamic representation of a system-to-be [3], [4]. Tool support for simulating and testing a combination of data and dynamic aspects of conceptual models for checking a model's conformance to the requirements it captures, is still limited.

The research presented in this paper addresses the issue of assisting the modeling process by providing modelers with tooling that enables them to test a model. The testing is achieved by 1. transforming a conceptual model to a prototype tailored to the technical expertise of a novice modeler and 2. incorporated cognitive feedbacks in this prototype application. The proposed approach aims to improve the learning outcomes for novice modelers with respect to modeling knowledge.

We will thus formulate our research question as *how can prototyping be helpful in teaching conceptual modeling, and how can the shortcomings of current simulation techniques be addressed to make a prototyping method more effective in a learning context?*

To assess the effects of prototyping on model quality parameters we will refer to the quality dimensions of the CMQF framework [2]. Within this framework, teaching conceptual modeling involves different types of modeling quality. The final objective is to achieve the capability of producing physical models with high semantic quality. Semantic quality refers to the "meaning" of a model which includes both its validity in terms of its correctness of representation of the domain at hand, as well as its completeness. On the knowledge side, this requires an appropriate level of model knowledge, language knowledge and representation knowledge, hence requiring pedagogical quality (understanding the modeling concepts), linguistic quality (understanding the graphical notation) and pragmatic quality (understanding a model)[2]. In particular, pragmatic quality captures the extent to which the stakeholder completely and accurately understands the statements in the representation that are relevant to them.

This work proposes an extension to the techniques previously presented by Snoeck et al. [5, 6, 7, 8]. The methodology used is based on the concepts of MERODE¹. The proposed MDA-based prototyping method has been tested and validated within the course "Architecture and Modeling of Management Information Systems"² over a 5-years period of teaching, with participation and constant feedback from 400 students overall.

The remainder of the paper is structured as follows. The second section gives an overview of related work concerning factors that affect teaching conceptual modeling and discusses learning perspectives with Model Driven Architecture (MDA). Section 3 outlines the proposed tool: a feedback-enabled MDA prototyping method to support the validation cycle for a conceptual model. Section 4 reports on the effects of experimenting with such prototypes on learning outcomes for novice modelers, and in particular on the extent to which they completely and accurately understand the statements in the model. Finally, section 5 concludes the work proposing some future research directions.

¹ MERODE is an Object Oriented Enterprise Modeling method. Its name is the abbreviation of Model driven, Existence dependency Relation, Object oriented DEvelopment. Cfr. <http://merode.econ.kuleuven.be>

² The course's page can be found on
<http://onderwijsaanbod.kuleuven.be/syllabi/e/D0I71AE.htm>

2 Related Work

Currently the learning process of novice modelers rarely includes (formal) quality checking techniques, and the tool support for integrating requirements and models is still limited [9]. The guidance on how to learn modeling is lacking and there seems to be very little discussion on how modeling skills might be taught [2]. Furthermore, there are aspects that cannot be gained with reading or lecturing alone, e.g. the ability to mentally transform a static model into a more concrete and dynamic representation of a system-to-be. With a traditional development process it is almost impossible to construct precise specifications. Hence, requirements satisfaction can only be established when the complete system has been built and the system can be examined in its most concrete form [10]. Despite being used for a long time, models in traditional approaches have been viewed as secondary artifacts and are in practice often abandoned [9]. Tool support for checking a model is limited to internal consistency and completeness of a conceptual model. Support to ensure the external validity of a model, i.e. how accurately and completely a model captures the domain, within the constraints of the modeling task at hand, is limited.

2.1 Prototyping with MDA

Simulation has been recognized by various studies as a technique that helps learning the dynamic aspect of a model, thereby improving the feedback cycles, and such technique contributes to better external validity [3], [4]. However tool support for simulating and testing a combination of data and dynamic aspects for conceptual models in order to check a model's conformance to the requirements it captures, is still limited. Several disadvantages for simulation techniques have been reported as well, such as: 1. being time-consuming to achieve; 2. requiring technical expertise not easily achievable by a novice user; 3. sometimes it is hard to analyze the results with a simulated model [11].

MERODE follows the Model-Driven Architecture and Engineering approach. Model-Driven Architecture (MDA) [12] refers to *model-centric* approach where models are essential parts of the final product [9]. MDA allows to transform a model into its executable form within the shortest development cycle allowing to validate static system designs by testing the (generated) working application. As such MDA represents a fundamental evolution with the potential to bring *design and testing* closer to each other. It is exactly in the testing area, with the validation of a working application, where its potential is the most significant [13]. We further rely on MDA since a model-centric process forces a set of *quality* parameters for a model such as being precise enough, formal enough, and detailed enough to generate code from it [13]. MDA also allows to enhance the understanding of the rationale behind system design decisions [9]: it is also concerned with uncovering the implicit knowledge that modelers rely on when designing models [9] since change propagations from requirements and models to the application become visible [13]. Such knowledge can be gained by tracing changes from a model to their effects in a working application by testing several “what-if” scenarios. When made explicit, such knowledge on how modeling options affect the final application can be further reused [9] thus increasing modeling expertise.

With MDA it is also possible to rely on modeling *process* oriented assistance. While many studies focus on the quality of a conceptual model, some approaches focus on the conceptual *modeling process* as a catalyst of a quality of its end result [2]. Comparing a modeling process between experts and novices [14] identifies ways in which novice modelers fall short of what is considered to be a good practice by experts: observing modeling process iteration cycles [14] distinguishes frequent switches between modeling and assessing among expert modelers (with twice as much time distributed on verification and validation activities by experts), while novice users' modeling behavior is quite linear (one task at a time) and more data-reliant. We believe that prototyping with MDA also allows to "mimic" an expert's modeling approach by making a modeling process of a novice modeler more verification-oriented.

Thus, by using an MDA simulation approach, the learning process of novice modelers can largely benefit from 1. familiarizing them with a *model-centric* approach of working, 2. increasing awareness of the *quality* parameters affecting the quality of a final artifact.

2.2 Requirements for Optimal Prototyping Environment

The efficiency of a prototyping environment in a learning context is associated with a set of requirements. In the previous years we observed that the prototyping tool was not extensively used by students. This is because the simulation was time-consuming to achieve because of requiring a chain of several transformation and execution steps before a running prototype was accessible. Another observed reason behind the low usage intention was the low technical expertise of novice modelers resulting in difficulties in managing these execution steps. This suggested that a prototyping environment would require a set of simplifications to be accepted and frequently used by novice learners. The well-known Technology Acceptance Model (TAM) [15] measures users' motivation by attitudes and intentions expressed with the two technology acceptance measures - *ease of use*, and *usefulness* by defining 1. the *perceived usefulness* (PU) as the degree to which a person believes that using a particular system would enhance his or her performance, and 2. the *perceived ease-of-use* (PEU) as the degree to which a person believes that using a particular system would be free from effort. With respect to the PEU for a novice modeler one of the major requirements is that the prototyping environment is adapted to a minimal technical expertise to allow easy and intuitive interaction with a computer. With respect to the PU a generated prototype should be capable to improve model understanding and subsequently the achievements of a novice modeler to stimulate a motivational engagement. Studies on learning quality improvements indicate a self-regulative approach as major source of impact on learning outcomes which in turn is closely intertwined with feedback research [16, 17]. Furthermore, for all self-regulative activities, external feedback is considered as an inherent catalyst [18]. Usually this feedback is not available during learning activities but is given after a task has been completed. This is referred to as outcome feedback, the simplest form of feedback, indicating whether or not results are correct, thus providing minimal external guidance [19]. [19] further specifies the

need for more informative types of feedbacks paired with content-related information: 1. cognitive validity feedback, describing a learner's perceptions about the relationship between a cue and achievement, 2. functional validity feedback, in general describing the relation between the learner's estimates of achievement and his or her actual performance. This suggests that combining prototyping with informative feedbacks incorporated in the prototype will result in high PU and subsequently high motivation among novice modelers to rely on validation cycles with a prototype application of a model.

Yet another important requirement for conceptual model prototyping is the capability to execute *platform independent models*, i.e. models with *high level of abstraction*.

Although UML aims at genericity by supporting modeling various views of a system, it on the other hand fails to provide good means of separating aspects per development phase (e.g. conceptual modeling versus program design). UML also fails to provide good support for recombining different views into one global and consistent model [20]. Executable UML (xUML) is the current standard for model execution; it is a profile of UML 2.0 to define the execution semantics for a subset of the UML. Recently, there was the introduction of the Foundational UML (fUML) [21] and the Action Language for fUML (Alf) [22], the new executable UML standards: fUML specifies precise semantics for an executable subset of UML, and Alf specifies a textual action language with fUML semantics. With these new standards, UML models can be executed. However, these techniques compromise the conceptual modeling character of the requirements (i.e. their high abstraction level) by requiring lower-level, detailed models [13, 21, 22]: e.g. by requiring behavior details such as actions and flows both in graphical and textual implementation (close to a Java programming language like syntax) in order to make models detailed enough for execution. These techniques are therefore technically too complex to be used as conceptual model specification techniques in an educational context.

The contribution of this work is to address the shortcomings of current conceptual model (MDA-based) simulation techniques being 1. time-consuming to achieve; 2. technically complex to achieve with the current standards for model execution; 3. providing no means to interpret the simulation results. We thus introduce a set of optimal simplifications: namely, 1. methods are offered and implemented for MDA-based prototyping cycle to be maximally tailored to minimal technical expertise, allowing an easily achievable simulation of conceptual models within the shortest cycles of time; 2. the results of simulation to be easily interpretable by incorporating automated cognitive feedbacks.

3 Implementation

3.1 Modeling Environment

In a previous paper [2] we already discussed a set of issues related to the widely accepted standard UML [20, 23, 24, 25, 26]. In its current form UML performs poor in supporting an educational perspective. UML also underperforms in terms of requiring a large technical skill-set to put MDE into practice with the current state-of-the-art of

MDE standards, thus being too complex in an educational context. UML thus requires a set of simplifications to be better tailored to novice modelers both for conceptual modeling as well as MDA-based simulation purposes. In this paper we will briefly outline those issues for readability purposes, however we remind that this work is an extension to [5] and the details of addressing UML complexity, namely the motivation of the restriction to a subset of UML and to particular views will be out of scope of the current paper.

To address the indicated UML issues MERODE adapts the use of UML to 1) alleviate the problem of "noisiness" of UML, and 2) ensure the quality and "transformability" of the model for model-driven architecture. In MERODE the object-oriented business model typically consists of 3 system views that together define a platform independent model. By removing or hiding details irrelevant for a conceptual modeling view makes the approach easier to understand. The business domain model in MERODE consists of a class diagram, an interaction model and a number of state charts. JMermaid is an adapted modeling tool for modeling conceptual business models based on the MERODE concepts. The 3 system views in the tool are represented with a tabbed view which suggests an intuitive, incremental and iterative modeling process. The class diagram is a restricted form of UML class diagram [27]: the types of associations are limited to binary associations, with a cardinality of 1 to many or 1 to 1. Many to many associations need to be converted to an intermediate class. The interaction model consists of an Object-Event Table (OET), created according to the principles of MERODE [1]. It specifies atomic transactions that potentially affects more than one class and requires a number of preconditions to be satisfied before execution.

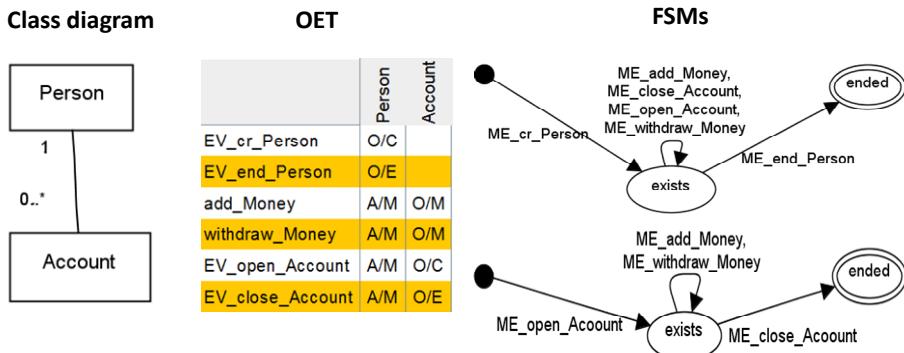


Fig. 1. Modeling views within JMermaid: Class diagram, Object-Event Table (OET) and Finite State Machines (FSMs)

Fig. 1 shows a snapshot combining the three main views supported in the JMermaid modeling tool. To ensure the completeness of a model to be processed by a code generator the tool uses consistency checking and validation techniques. To simplify its usage, the tool allows managing consistency between the three views in an

automated way: it follows a "consistency-by-construction" approach [7, 8] meaning that each time when entering specifications in one view, specifications that can be derived for other views are automatically generated by the tool. As an example, one of the design guidelines states that when defining a class, one should provide at least one method to create instances of that class and one method to terminate instances. So when a business object is entered in the class diagram, the necessary completions are automatically performed in the OET and FSM views. This modeling approach ensures a perfect integration between the structural, interactive, and behavioral aspects, achieving models that are truly executable to be further validated through the prototyping feature.

3.2 MDA-Based Prototyping Environment

To conform to the requirements suggested by the Technology Acceptance Model MERODE maximally tailors the prototyping environment to the technical expertise of a novice modeler. Transformation to code can be achieved through a single click. The MDA-based code generator generates a fully functional application, namely, a compiled application in executable JAR format. With such simplification of the transformation process testing of a model can be started in parallel with model building. The minimal input that can be accepted by the prototyping tool is actually a model that contains at least one business object in the class diagram view along with the minimal set of default elements, state machine states and transitions that are automatically generated by JMermaid. A set of default attributes for business objects, if not specified by a user, are automatically generated too.

A modeler can use the "Verify model" functionality in JMermaid to detect internal inconsistencies [7, 8] such as missing parts (e.g. missing creating or ending events), inconsistencies in entities' lifecycles (FSM) such as unused methods, etc. Additionally this verification is referred to from the prototyping tool, e.g. each time a compilation of the generated code fails a student gets a message to use the "Verify model" functionality to ensure the completeness of a model. Such a loop can increase modeling skills with respect to quality requirements for a conceptual model.

The layout of a prototype application is quite intuitive. The graphical user interface of the prototype offers basic functionality like triggering the creating and ending of objects, and triggering other business events. Fig. 2 shows the main interface of a generated prototype. The GUI layer is built on top of the event handling layer. The event handling layer consists of a collection of so called event handlers. The task of the latter layer is to handle all events correctly by managing the appropriate interactions with the objects in the persistence layer.

The working of an event handler can be described in four steps: 1) upon an event execution call the event handler 'asks' every participating object (the participants to a business event that have been specified in the Object-Event Table) whether all preconditions set by the object are met.

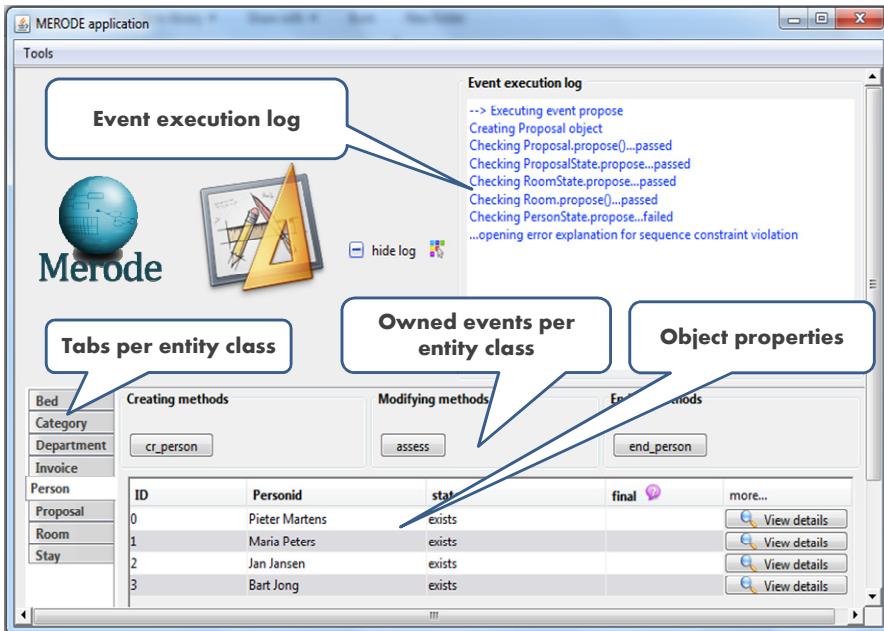


Fig. 2. Main user interface of a prototype application

For example, associations between classes will lead to preconditions to maintain referential integrity; 2) Similarly to the previous step the event handler retrieves from every participating object its current state (or reference to the corresponding state object) and checks whether that state allows further processing of the event; 3) If all results of the tasks in step 1 and 2 are positive, the event handler invokes the methods in the participating objects, i.e. corresponding event triggered in response to processing the originally called event in the specific object; 4) next, if all results of previous steps are positive, the event handler executes the method in all participating objects retrieved in step 2 to implement the state modifications (according to the triggered event). While executing a business event in a prototype application users can follow in an event execution log frame what is happening in the upper right corner of the generated application.

3.3 Feedback-Enabled Testing Cycle

The prototype application is augmented with a set of cognitive feedbacks to provide increased transparency between the model and its prototype [5] reporting a user about illegal calls of events with informative messages and graphical visualizations, e.g. when an event is refused (because of failed precondition checks) the user is informed of the refusal with a message that explains the reason of rejection by indicating what constraint of a model is violated (e.g. referential integrity constraint, creation/end dependency or integrity constraint, FSM imposed constraint, etc.). Fig. 3 shows for example how the triggering of a business event is refused by the application because the business rules stated in the form of a Finite State Chart impose a precondition that is not met by the current state of the business objects.

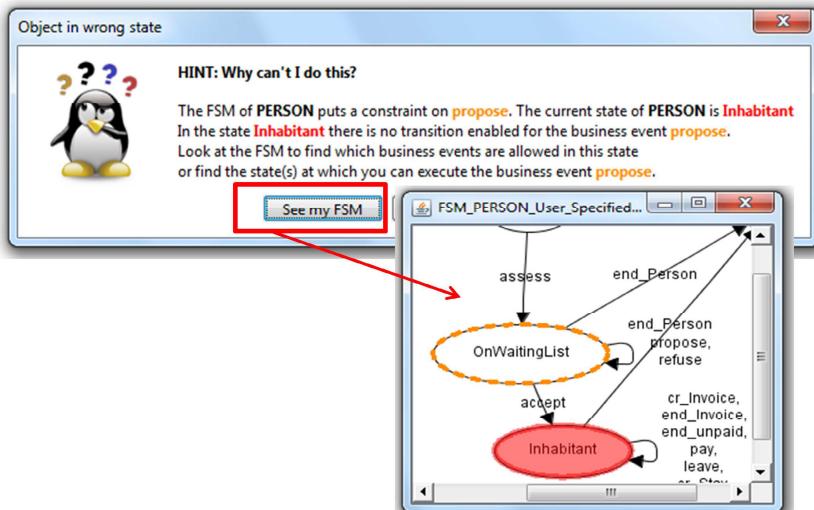


Fig. 3. Example of an automated feedback to an event execution failure

Such model execution with automated feedback enables a much better understanding of models than can be obtained by just reading a model, hence motivating a frequent use with the purpose to detect and correct errors while also gaining knowledge on the modeling method and modeling language.

4 Evaluation and Experiences

To assess the effects of a feedback-enabled MDA-prototyping with respect to model understanding aspects, we conducted an experiment with participation of 49 students. Students were given a short requirements statement and a model solution. Students were asked to evaluate the quality parameters of a model by responding to a set of true/false format questions validating whether or not the provided model meets the given business requirements, hence testing their model understanding. They were also asked to motivate their answers. The experiment was conducted in two parts. In the first part the experiment was conducted with a static model (a UML Class Diagram) without a use of the model's prototype. The goal of this part was to establish a baseline knowledge level to measure the effects of the suggested prototyping method in a later experiment. Then, in the second part of the experiment, the same questions had to be answered again with the use of generated prototype, however being unassisted in the testing process, i.e. not being given any testing scenarios. Blank answers as well as those without motivation have been truncated to eliminate false negative and false positive corrections, resulting in 26 usable answer sheets. Table 1 shows the numbers of correct answers for students obtained before and after the simulation cycle along with the correction effect after validating a model with its prototype.

Table 1. Assessment of model understanding before/after a prototype use

	Nr of correct answers (out of 26)		Percent (%)		Correction		
	Q	(control) before	(experimental) after	before	after	Nr corrected	%
1		9	24	34,6%	92,3%	15	57,7%
2		15	23	57,7%	88,5%	8	30,8%
3		19	24	73,1%	92,3%	5	19,2%
4		11	20	42,3%	76,9%	9	34,6%
5		6	8	23,1%	30,8%	2	7,7%
6		11	15	42,3%	57,7%	4	15,4%
7		7	19	26,9%	73,1%	12	46,2%
8		6	5	23,1%	19,2%	-1	-3,8%
9		1	15	3,8%	57,7%	14	53,8%
	AVG = 9,44		AVG = 17			AVG DIFF = 7,55	

The results show a significant correction for the student's model understanding after the use of a prototype with positive correction in scores ranging from 7,7% up to 57,7%. A paired t-test was performed to determine if the method was effective: Hypothesis H1 M-after = M-before, alternative hypothesis H2 M-after > M-before. The mean correction gain ($M = 7.55$, $N = 9$) was significantly greater than zero (thus refuting H1 and supporting H2), t-stat = 4.1193, 1-tail $p = 0.00167$, with 95% confidence interval providing evidence that the prototyping method is effective in producing positive correction of model understanding.

Students from different programs were enrolled in the course including those with bachelor degree in Information Systems. Since these students have previous knowledge and familiarity with a modeling method such as UML we applied a clustering approach to observe the outcomes for students with and without prior modeling skills. The results however showed equal correction for both groups. For a particular question the experiment however resulted in a negative correction. We observed this to be due to a lack of testing skills to identify right scenarios.

A second validation was performed using the coursework. In the course of the semester, students have to create a model for a larger case (typically 2-5 pages of specifications, resulting in a model with around 15 domain objects). At the end of the semester the solution is scored, and then students are interrogated to determine the final score as a correction on the model score. The goal of this oral interrogation is to test whether a student 1) truly understands his/her own solution 2) is able to see mistakes identified by the professor and 3) is able to suggest a solution for the mistake. In this case, testing assistance is provided as students are given a scenario by the teacher that specifically aims at discovering certain model errors. For example, if the case specifies that a library member can borrow at most 3 books at a time, then the teacher provides a scenario in which a member attempts to borrow 4 books at once. The answer of the student is scored as a correction on the score of the proposed solution. If the student doesn't understand that the proposed solution does not satisfy the requirements, a negative score is given. If the student can locate the problem but doesn't know how to correct it, a 0 is given (i.e. score of the solution is maintained). If the student can locate the problem and propose a correction, a positive score is attributed.

In January 2012 students were asked to demonstrate their solution by manually simulating the model using a test case provided by the teacher. Overall, the manual testing led to a negative correction on the proposed solution (see Table 2). More than half of the students were not able to identify mistakes in their solution, not even when simulating it with a given test scenario, leading to on average a decrease of 0.19 of the score obtained by scoring the proposed model only. When provided with a test scenario, students that lack model understanding, will tell a story based on the scenario and the original requirements, as if the model captures the requirements correctly. A typical example is modeling the requirement "Orders can be placed by private customers or business customers" as in Fig. 4. The students interpreted the two associations as alternatives, whereas both associations exist simultaneously and hence -according to the model- each order is placed by both a Business Customer *and* a Private Customer. Students' incapacities of reading a model leads to an incorrect evaluation of a test scenario.

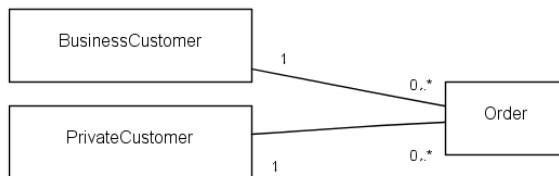


Fig. 4. Orders and Customers model

In January 2013, the same type of evaluation was performed, but this time students had to *execute* the given test scenario using the prototype. By means of the dynamic testing approach almost half of the students were able to see and correct mistakes. Overall, a positive correction of 0.19 points (on 10) was observed. Although this result is positive, we nevertheless observed student incapacities to develop their own adequate test scenarios. The 2013 case was about ordering and delivering baskets of vegetables. The teacher proposed a test scenario in which an order for cauliflowers was registered, but the picking was wrongly registered as tomatoes. Some students would simply answer "impossible to register because no tomatoes were ever registered as type of vegetable", whereas better students would first create tomatoes as type of vegetable in the system and then test if the wrong registration is captured or not by means of modeled constraints.

Hence, two conclusions are obtained: 1. the results of the experiment and the oral examination demonstrate that the validation by means of a working prototype improves modeling understanding compared to a paper exercise. The paper exercises limit the scope of understanding to a static view of a model, whereas dynamic testing fosters a more thorough understanding; 2. Validation cycles supported with test scenarios provided by the teacher resulted in better model understanding indicators than unassisted testing cycles.

Table 2. Correction of student courseworks for 2012 (manual simulation) and 2013 (prototype execution)

	2012		2013	
Value of total average correction (on 10)	-0.19		0.19	
Number of students making a positive correction	27	34,6 %	29	43,3 %
Number of students making a negative correction or not understanding their own model	36	46,2%	16	23,9 %
Number of students understanding their model, but not able to correct errors	15	19,2%	22	32,8%

Next, a questionnaire method was used to collect student feedback on the perceived usefulness of a method with respect to their learning achievements on a range of 1 (not useful) to 5 (very useful). Table 3 shows the results of the evaluation for three successive years of teaching based on the answers of 150 participated students (approximately 50 per year). The empty cells indicate that the feature was not available at the time of evaluation.

Table 3. Evaluation of prototyping and feedback effects by students

<i>How helpful were the following features ? (1 = Not helpful, ..., 5 = very helpful)</i>	2011	2012	2013
outcome feedback	informative feedback (WHY)	informative feedbacks + prototype tailored to technical expertise for novices	
Code Generator / Rapid Prototyping	3,7	3,46	4,58
Explanation and graphical visualizations of errors in prototype	-	3,87	4,52

Until 2011 the simulation was achieved through a chain of several transformation and execution steps before being able to run the prototype. Violations of validation rules by illegal calls of business events in a prototype were initially reported to students in the form of error popups (e.g. “Can’t execute event: object in wrong state.”) which were not easily interpreted by students. Because of low technical expertise the prototyping feature was not extensively used by students subsequently being scored 3,7 on average by students.

In 2012 error explanations and graphical visualizations were implemented as an optional plugin students could extend their prototypes with. Although the plugin scored slightly higher (3,87), due to their low technical skills students still experienced various difficulties throughout the simulation process chain, first with generating a prototype, and next extending it with a plugin. This made the major part of students reluctant in using the feature mostly resulting in “didn’t use” answer while evaluating the feature. In the meantime, the problems with the simulation chain have been solved

by providing students with an all-in-one package allowing to generate and start a prototype with a single click from the student side (2013) thus solving the issue of the simulation cycle being time-consuming to achieve, as well as being often avoided due to a high technical expertise required to achieve such simulation. Evaluation by students resulted in 4,58 for the prototyping tool and 4,52 for the incorporated feedbacks.

The results suggest that the combination of the cognitive feedbacks and the ease-of-use of a simulation tool can stimulate motivation and frequent validation cycles among students. It is however impossible from our experiments to distinguish the proportional impact of feedbacks on the one hand and from the prototype on the other hand. Since the validity of these results are limited to the context of the course AMIS and 26 students, we plan future experiments with a larger sample and multiple cases/models varying in complexity levels to obtain better accuracy and insights. The variation of positive effect from question to question within a range of 7,7% to 57,7% needs to be further analyzed with a better experiment in order to define the exact scope of effects and possible improvements along those dimensions where a smaller significance is achieved. Improved experiments with other courses/universities as well as industry users are therefore planned.

5 Conclusion and Future Work

The feedback-enabled MDA prototyping does have a great impact on improving modeling knowledge (as well as a modeling language knowledge). This improved knowledge enable students to better understand a model, contributing in this way to what is called pragmatic quality in [2]. At the same time this also enables a student to better assess the semantic quality of a model, in terms of his/her ability to judge the validity of a model with respect to its correctness of representation of the domain at hand. Tool support allowing generating a working application out of a conceptual model was observed to have a self-regulative effect stimulating a motivational use of a prototype to better understand models, detect errors, revisit and refine models by novice modelers. We hence witness an increased pragmatic quality, followed subsequently by a higher achieved semantic quality of the models. The tool support makes the time distribution in the modeling process closer to those of expert modelers with respect to time devoted to validation activities as well as increasing engagement in modeling activities in general resulting in better timing for training. The improvements among novice modelers are observed to be equal both for those having a prior knowledge of a modeling method/language and those without a prior knowledge. However some improvements and a set of simplifications can be further applied. Among possible directions for further research we consider:

1. Adapting the graphical notation to conceptual modeling goals. This is motivated by the fact that several studies indicate graphical notation having its own share in decreasing requirements externalization quality among novice modelers [28, 29].

2. Stimulating testing skills of a novice modeler in parallel with modeling knowledge acquisition due to the low level of testing capabilities observed among novice modelers. Tool support can be investigated to enable automated generation of test scenarios out of textual requirements description to assist in testing a simulated model against business requirements.
3. Designing experiments to obtain better understanding of modeling approaches used by expert and novice modeler, e.g. modeling behavior patterns such as time distribution on various tasks, validation cycle frequency effects with the use of a simulated model.
4. Expanding visual aids, e.g. visualizing propagation effects for a model modification, i.e. parts of a model that will be affected once a change is made.
5. Investigating ways of providing automated NLP assistance for the requirements analysis phase.
6. Based on the observations expanding the scope of computer-assisted cognitive feedbacks throughout a modeling process for novice modelers.
7. Extension with an ability to generate code from models that use inheritance and support for general constraints formulated in OCL.

References

1. Snoeck, M., Dedene, G., Verhelst, M., Depuydt, A.: Object-oriented enterprise modelling with MERODE. Leuvense Universitaire Pers, Leuven (1999)
2. Nelson, H.J., Poels, G., Genero, M., Piattini, M.: A conceptual modeling quality framework. *Software Qual. J.* 20, 201–228 (2012)
3. Merrill, D., Collofello, J.S.: Improving Software Project Management Skills Using a Software Project Simulator. In: Frontiers in Education Conference (1997)
4. Neu, H., Becker-Kornstaedt, U.: Learning and Understanding a Software Process through Simulation of Its Underlying Model. In: Proceedings of LSO, pp. 81–93 (2002)
5. Sedrakyan, G., Snoeck, M.: Technology-enhanced support for learning conceptual modeling. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (eds.) BPMDS 2012 and EMMSAD 2012. LNBP, vol. 113, pp. 435–449. Springer, Heidelberg (2012)
6. Snoeck, M., Haesen, R., Buelens, H., De Backer, M., Monsieur, G.: Computer Aided Modelling Exercises. *Journal Informatics in Education* 6(1), 231–248 (2007)
7. Snoeck, M., Michiels, C., Dedene, G.: Consistency by construction: The case of MERODE. In: Jeusfeld, M.A., Pastor, Ó. (eds.) ER Workshops 2003. LNCS, vol. 2814, pp. 105–117. Springer, Heidelberg (2003)
8. Haesen, R., Snoeck, M.: Implementing Consistency Management Techniques for Conceptual Modeling. Accepted for UML 2004: 7th Conference in the UML Series, Lisbon, Portugal, October 10-15 (2004)
9. Zikra, I., Stirna, J., Zdravkovic, J.: Analyzing the Integration between Requirements and Models in Model Driven Development. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBP, vol. 81, pp. 342–356. Springer, Heidelberg (2011)
10. Ince, D.C., Hekmatpour, S.: Software prototyping – progress and prospects. *Information and Software Technology* 29(1), 8–14 (1987)

11. Klingstam, P., Gullander, P.: Overview of simulation tools for computer-aided production engineering. *Computers in Industry* 38, 173–186 (1999)
12. OMG, Model-Driven Architecture, <http://www.omg.org/mda/>
13. Conn, S.S., Forrester, L.: Model Driven Architecture: A Research Review for Information Systems Educators Teaching Software Development. *Information Systems Education Journal* 4(43) (2006)
14. Wang, W., Brooks, R.J.: Empirical investigations of conceptual modeling and the modeling process. In: Henderson, S.G., Biller, B., Hsieh, M.H. (eds.) *Proceedings of the 39th Conference on Winter Simulation*, pp. 762–770. IEEE, Washington, DC (2007)
15. Davis, F.D., Bagozzi, R.P., Warshaw, P.R.: User acceptance of computer technology: A comparison of two theoretical models. *Management Science* 35, 982–1003 (1989)
16. Zimmerman, B.J.: Investigating Self-Regulation and Motivation: Historical Background, Methodological Developments, and Future Prospects. *American Educational Research Journal* 45, 166–183 (2008)
17. Nicol, D.J., Macfarlane-Dick, D.: Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in Higher Education* 31(2), 199–218 (2006)
18. Barber, L.K., Babsby, P.G., Grawitch, M.J., Buerck, J.P.: Facilitating self-regulated learning with technology: Evidence for student motivation and exam improvement. *Teaching of Psychology* (2011)
19. Butthler, D.L., Winne, P.H.: Feedback and Self-Regulated Learning: A Theoretical Synthesis. *Review of Educational Research* 65, 245–281 (1995)
20. Buckl, S., Matthes, F., Schweda, C.M.: A Meta-language for EA Information Modeling - State-of-the-Art and Requirements Elicitation. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) *BPMDS 2010 and EMMSAD 2010*. LNBP, vol. 50, pp. 169–181. Springer, Heidelberg (2010)
21. Foundational Subset for Executable UML Models (FUML), Version 1.0., <http://www.omg.org/spec/FUML/1.0/>
22. Alf 1.0 Specification, <http://www.omg.org/spec/ALF/Current>
23. Siau, K., Loo, P.-P.: Identifying difficulties in learning UML. *Information Systems Management* 23(3), 43–51 (2006)
24. Moisan, S., Rigault, J.-P.: Teaching Object-Oriented Modeling and UML to Various Audiences. In: Ghosh, S. (ed.) *MODELS 2009*. LNCS, vol. 6002, pp. 40–54. Springer, Heidelberg (2010)
25. Erickson, J., Siau, K.: Can UML Be Simplified? Practitioner Use of UML in Separate Domains. In: *Proceedings EMMSAD 2007*, Trondheim, Norway, pp. 87–96 (2007)
26. Gustas, R.: Conceptual Modeling and Integration of Static and Dynamic Aspects of Service Architectures. In: Sicilia, M.-A., Kop, C., Sartori, F. (eds.) *ONTOSE 2010*. LNBP, vol. 62, pp. 17–32. Springer, Heidelberg (2010)
27. Snoeck, M., Dedene, G.: Existence dependency: the key to semantic integrity between structural and behavioural aspects of object types. *IEEE Trans. Software Eng.* 24(4), 233–251 (1998)
28. Recker, J., Safrudin, N., Rosemann, M.: How Novices Model Business Processes. In: Hull, R., Mendling, J., Tai, S. (eds.) *BPM 2010*. LNCS, vol. 6336, pp. 29–44. Springer, Heidelberg (2010)
29. Moody, D.L.: The “physics” of notations: a scientific approach to designing visual notations in software engineering. In: *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, New York, NY, USA, vol. 2, pp. 485–486 (2010)

A Rigorous Reasoning about Model Transformations Using the B Method

Akram Idani¹, Yves Ledru¹, and Adil Anwar²

¹ Université Joseph Fourier, Grenoble INP, CNRS
LIG, UMR5217, 681 Rue de la Passerelle (BP 72)
38402 Saint Martin d'Hères Cedex

{Akram.Idani,Yves.Ledru}@imag.fr

² Siweb, Ecole Mohammadia d'Ingénieurs
Mohamed Vth Agdal University
Avenue Ibnsina B.P. 765 Rabat, Morocco
anwar@emi.ac.ma

Abstract. A crucial idea of Model Driven Engineering is that model transformation can be described uniformly in terms of meta-model mappings. Based on the fact that meta-models define an abstract syntax from which one can describe elements of modeling languages, transformation rules that arise from MDA-based techniques are often described as explicit and clear. However, one of the remaining difficulties is to check the correctness of these transformations in order to prove that they preserve constraints which may be expressed over meta-models. Currently, the MDE gives methodological issues for the use of OCL to express these constraints but without providing automated formal reasonings. This paper discusses how a formal method, such as B, can be used in an MDE process in order to rigorously reason about meta-models and associated model transformations. We propose to adapt existing UML-to-B techniques in order to obtain a formal specification of meta-models and hence the various constraints can be introduced using B invariants. We also show how transformation rules can be encoded using B operations and what kinds of reasoning can be performed on the resulting B specifications. Such a technique allows to assist the MDE by proof and animation tools.

Keywords: Model Transformation, Meta-Models, B Method, Method integration, UML-to-B.

1 Introduction

The Model Driven Engineering (or MDE) is an iterative development approach in which the software development process is based on a set of step-by-step refinements (or integration) of models. It distinguishes platform independent models (PIM) and platform specific models (PSM). The development process is hence seen as a gradual transformation of a PIM model, which specifies a business solution independently of the target technologies, to a PSM model which describes how this solution can be implemented. Platforms that support this approach

(*e.g.* [4, 5]) require meta-models as a description of the manipulated models. The PIM-to-PSM transformation rules are then expressed by a set of mappings from a source meta-model to a target meta-model. Based on the OMG standards [14], these tools have reached a good level of maturity. However, although they seem to be useful for safety-critical systems, safety challenges about systems that they produce are still open. Indeed, the existing MDE platforms are focused on the usability of meta-models and transformation rules, but without offering a way to prove their correctness.

In the existing MDE platforms, the common way to validate MDE artifacts is to test the transformation rules on a set of existing input models. This a posteriori validation allows to check if the produced PSM model conforms to its meta-model. For an a priori validation, mathematical languages, clearly defined with precise semantics and which allow proofs must be used. Unfortunately, this is not the case of the OMG standards (MOF and QVT) because they are based on graphical notations which don't offer proof tools. In order to circumvent this shortcoming, we propose to bring the MDE to the rigorous world of the B formal method [1]. Our goals are:

1. To perform automated formal reasonings, using a prover, when designing the PIM-to-PSM transformation. The objective is to formally validate the transformation rules and hence cover the a priori validation.
2. To simulate the transformations using a B animator (*e.g.* ProB [10], BZ-TT [9], ...). The objective is to generate the target models from a set of source models and hence cover the a posteriori validation.

This paper discusses how a formal method, such as B, can be used in an MDE process in order to rigorously reason about meta-models and associated model transformations. We propose to adapt existing UML-to-B techniques in order to obtain a formal specification of meta-models and hence the various constraints can be introduced using B invariants. We also show how transformation rules can be encoded using B operations and what kinds of reasoning can be performed on the resulting B specifications.

This paper is organized as follows: section 2 gives a simple example to guide our proposal. In section 3 we show how UML-to-B approaches can be adapted in order to translate meta-models into B. Section 4 addresses the proposed proof-based MDE approach. Finally, section 5 gives the conclusion and the perspectives of this work.

2 A Simple Example

2.1 Meta-models

Let us consider a simple classical example dedicated to the transformation of oriented graphs into oriented multi-graphs. Figure 1 gives the corresponding meta-models. In an oriented graph (left hand side of figure 1), there is at most one edge between a given couple of nodes. This is specified by the association

predecessor/successor. Indeed, in the MOF an association can be instantiated only once between a couple of class instances. In order to be able to represent multi-graphs, it is necessary to introduce a class which can be instantiated several times. This is done by class *MEdge* in the right hand side of figure 1.

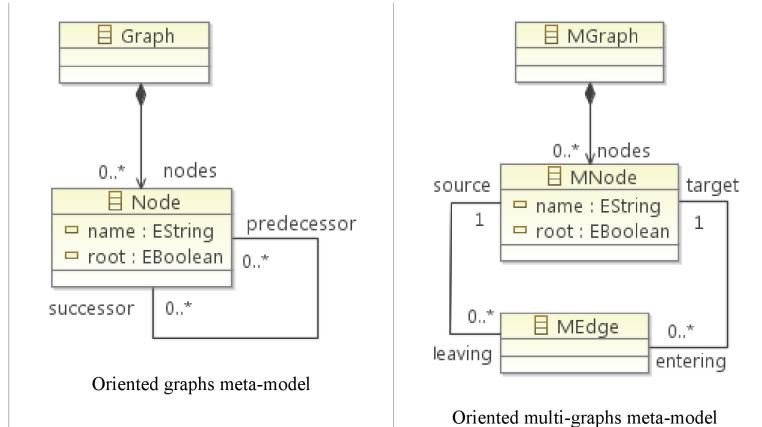


Fig. 1. Oriented graph and multi-graph meta-models

For this simple example, we impose that there is one and only one root node in a multi-graph using the `oneRootOutputModel` OCL constraint:

```
context MGraph inv oneRootOutputModel :
self.nodes->select(root=true)->size()=1
```

2.2 Transformation Rules

In order to illustrate how transformation rules can be encoded in a classical MDE platform, we chose ATL [3] which is a hybrid transformation language allowing to combine both declarative and imperative approaches. ATL is considered as a standard component of Eclipse for model transformation and is now integrated into the M2M project¹.

The following ATL rules are encoded in an intuitive manner in order to produce a multi-graph from a simple graph. Rule `translateGraph` below links classes `Graph` and `MGraph` and means that every instance of class `Graph` will produce an instance of class `MGraph` and the `nodes` linked to an `MGraph` are those issued from association `nodes` of a simple graph. This first rule takes each instance (called `grIn`) of class `Graph` in SimpleGraph meta-model (from `grIn : SimpleGraph!Graph`) in order to create an instance (called `grOut`) of class `MGraph` in MultiGraph meta-model (to `grOut : MultiGraph!MGraph`), and indicates that `grOut.nodes` is formed by the transformation of the elements of set `grIn.nodes`.

¹ Eclipse/M2M Project Web Page. <http://www.eclipse.org/m2m/>

```

rule translateGraph {
  from grIn : SimpleGraph!Graph
  to grOut : MultiGraph!MGraph (
    nodes <- grIn.nodes
  )
}

```

Rule `translateNode` links classes `Node` and `MNode` and means that every instance of class `Node` is translated into an instance of class `MNode`. This rule produces respectively the *leaving* and the *entering* relations. For each node `nodeIn` (from `nodeIn : SimpleGraph!Node`), the rule produces a node `nodeOut` (to `nodeOut : MultiGraph!MNode`) having the same values of attributes *name* and *root*. Then, collection *leaving* of `nodeOut` is formed by applying rule `translateEdge` on all elements of the collection `nodeIn.successor`. Collection *entering* of `nodeOut` is also formed by applying rule `translateEdge` on elements of the collection `nodeIn.predecessor`. This is intended to create links *leaving* and *entering* between the `nodeOut` and instances of `MEdge` issued from associations *predecessor* and *successor*.

```

rule translateNode {
  from nodeIn : SimpleGraph!Node
  to nodeOut : MultiGraph!MNode (
    name <- nodeIn.name,
    root <- nodeIn.root,
    leaving <- nodeIn.successor -> collect(ssSucc
        | translateEdge(nodeIn, ssSucc)),
    entering <- nodeIn.predecessor -> collect(ssPred
        | translateEdge(ssPred, nodeIn))
  )
}

```

Rule `translateEdge` takes two nodes `ssPred` and `ssSucc` which are assumed to be linked by relation predecessor/ancestor and produces an instance `tt` of class `MEdge` such that source and target of `tt` are respectively `ssPred` and `ssSucc`. It also produces an instance `tt` of class `MEdge` such that source and target of `tt` are respectively `ssSucc` and `ssPred`.

```

lazy rule translateEdge {
  from ssPred : SimpleGraph!Node, ssSucc : SimpleGraph!Node
  to tt : MultiGraph!MEdge (
    source <- ssPred,
    target <- ssSucc
  )
}

```

Note that in ATL a lazy Rule is a declarative rule which is explicitly called such as `translateEdge` which is called in rule `translateNode`. The other rules (`translateNode` and `translateGraph`) are called, in ATL, matched rules

because they are automatically triggered by the ATL engine when their “from” part is matched.

2.3 Execution

Figure 2 gives an example of input and output models of the ATL rules. The input model is a simple oriented graph containing nodes A and B and an edge oriented from A to B. The resulting model is a multi-graph in which two instances of class MEdge are produced. The multi-graph contains then nodes A and B and two edges oriented from A to B.

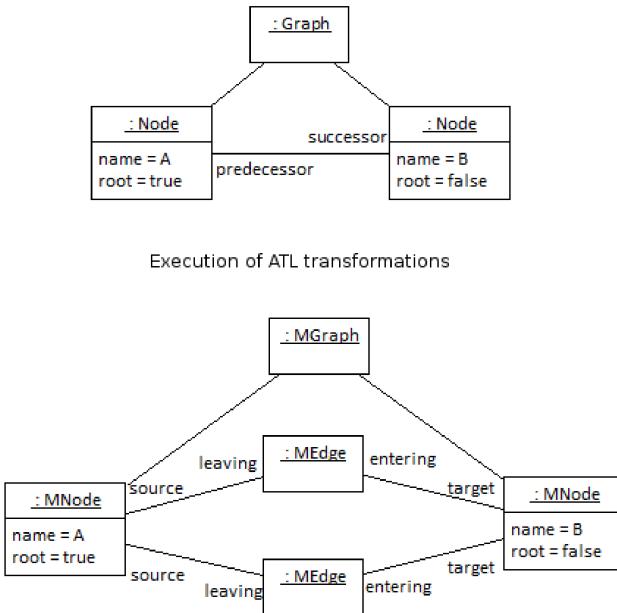


Fig. 2. Input and output models of the ATL rules

By observing the resulting model we can see that two edges are produced from one unique edge in the source model. Each edge is generated twice from the translation of predecessor/successor. This shows that it is of a great interest to express invariants that link both meta-models. For instance one should consider that the number of instances of class MEdge is equal to the number of links predecessor/successor of the input model. A formal method, such as B, allows to locate these defects in the transformation rules, to address them and to prove the rule correctness. Furthermore, this test case shows that this particular transformation produces a valid model which respects constraint `oneRootOutputModel`. Such a constraint can be easily evaluated by an OCL interpreter. However, it doesn't exhibit the properties of the transformations. Indeed, although test case of figure 2 is valid it can't attest that for each input model the transformation produces a valid output model. Indeed, if we try an input model which contains two root nodes, the ATL rules would produce two root nodes too and then

obtain an erroneous multi-graph. This can be avoided by adding a precondition to the transformation or an invariant to the source meta-model. Both kinds of properties can be easily addressed by a formal method such as B, as we will show in section 3.

3 UML-to-B for MDE

In order to assist the validation of model transformations we propose a proof-based technique, which is complementary to testing techniques [2]. Our proposal is inspired by the research works which tried to integrate UML and B [6, 13]. These works were motivated by the fact that UML diagrams are widely accepted as a standard for modeling software systems, but the lack of precise semantical basis results in a limited support for validating models early in the development process. The UML-to-B translation rules proposed by these approaches are intended to precisely define the semantics of the UML language, and to formally reason on the resulting B specifications using automated tools (provers and animators). Given that meta-models are kinds of class diagrams, and models are instances of these class diagrams, UML-to-B approaches can be adapted for a proof-based MDE approach that uses the B method.

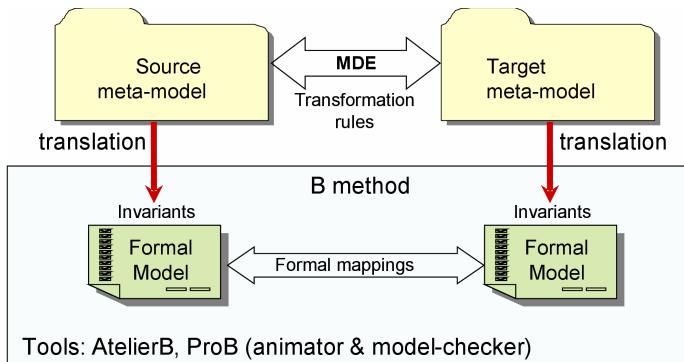


Fig. 3. Overall methodology

Figure 3 shows how our approach is situated in a classical MDE context. Meta-models are firstly translated into B. Then the transformation rules are encoded by a set of B operations. The resulting B specifications can be refined by taking into account, in the form of invariants, constraints of the input and output meta-models and also constraints on the links between these meta-models.

3.1 Translating Meta-models

Several research works proposed translation rules from a class diagram into B [11–13]. Then, the large number of class diagrams concepts which are addressed by these works motivate our work to adapt their techniques for all concepts of the MOF. Indeed, technically the MOF gives a subset of UML class diagrams (without associative classes, etc).

The following B machine named **SimpleGraphs** is derived from the input meta-model by applying principles of UML-to-B approaches.

MACHINE	PROPERTIES
SimpleGraphs SETS $ModelElement ;$ $NAMES$ CONSTANTS $Graph, Graph_nodes$ $Node, Node_name, Node_root,$ $predecessor_successor$	$Graph \subseteq ModelElement$ $\wedge Node \subseteq ModelElement$ $\wedge Node_name \in Node \rightarrow NAMES$ $\wedge Node_root \in Node \rightarrow \text{BOOL}$ $\wedge predecessor_successor \in Node \leftrightarrow Node$ $\wedge Graph_nodes \in Node \rightarrow Graph$

We specify the various model elements using constants because the source model is not modified by transformation rules. Classes are translated into sets which are included in an abstract set named **ModelElement**. Class attributes (*e.g.* *name* and *root*) are translated into partial relations (*e.g.* *Node_name* and *Node_root*). Associations are translated into specializations of B relations depending on multiplicities. For example an association with multiplicities * and 1 in its extremities leads to a total function (*e.g.* *Graph_nodes*).

The translation applied to the target meta-model is similar and results in the following structural part of machine **MultiGraphs**.

MACHINE	INVARIANT
MultiGraphs SEES <i>SimpleGraphs</i> SETS $MEDGE$ VARIABLES $MGraph, MGraph_nodes$ $MNode, MEdge,$ $MNode_name, MNode_root,$ $leaving_source, entering_target$	$MGraph \subseteq ModelElement$ $\wedge MNode \subseteq ModelElement$ $\wedge MEdge \subseteq MEDGE$ $\wedge leaving_source \in MEdge \rightarrow MNode$ $\wedge entering_target \in MEdge \rightarrow MNode$ $\wedge MNode_name \in MNode \rightarrow NAMES$ $\wedge MNode_root \in MNode \rightarrow \text{BOOL}$ $\wedge MGraph_nodes \in MNode \rightarrow MGraph$

Contrary to the input model, transformation rules are intended to modify the output model. That's why we use variables instead of constants. Meta-classes instances in the target model can be directly issued from meta-classes instances of the source model, but also newly created. In the previous ATL rules, instances of class **MGraph** and **MNode** are produced respectively from instances of classes **Graph** and **Node**. However, instances of classes **MEdge** are not issued from objects of the source model. Set **ModelElement** allows to group instances of meta-classes in the target model which are issued from instances of meta-classes of the source meta-model. For model elements which are newly created we consider an abstract set \mathcal{P} representing the set of possible instances of a model element (*e.g.* **MEDGES**) and a set \mathcal{E} representing the set of existing instances. The invariant $\mathcal{E} \subseteq \mathcal{P}$ links both sets. This translation is classical and widely discussed by the UML-to-B approaches. All other variables are included in the

abstract set `ModelElement` issued from machine *SimpleGraphs*. This choice is justified by the fact that these variables will be constructed from source model elements.

The mapping proposed here is different from principles of classical MDE tools because existing tools don't consider links between meta-models. For example, classes `Graph` and `MGraph` are different and associated to different meta-models. Hence there is no way to express in OCL that `Graph.AllInstances()` is equal to `MGraph.AllInstances()`. MDE tools consider that attribute `name` has unique values and then OCL expressions like `Graph.AllInstances() -> collect(name)` and `MGraph.AllInstances() -> collect(name)` are used. In our translation, we are guided by the transformation process which distinguishes, for the target model, elements which are identical to those of the source model and elements newly created. Target elements which are identical to those of the source model are included in the set of source `ModelElements`. This translation allows to type target elements by source elements using invariants such as: $MGraph \subseteq Graph$.

A natural limitation of the proposed translation is that it doesn't allow multiple transformations like transforming a simple graph into a multi-graph which will be transformed later into a tree etc. In order to remedy this limitation we can translate meta-models separately and links between them would be included in a third B machine. Both kinds of translations from meta-models into B are possible in our current works. For this paper, we choose the first kind because we deal with a quite simple example.

Discussion. Existing works which tried to formalize in B model transformations [7, 8] proposed their own formalization. They don't propose to adapt existing UML-to-B approaches, contrary to our proposal. In our work, our main intention is to be able to encode their translation principles from ecore meta-models into B in existing EMF-based platforms². Hence, the analyst reasoning, is not focused on formalization of meta-models but on associated transformation rules and constraints. Adaptation of UML-to-B approaches is useful in this context, because their concepts are generalized on several kinds of class diagrams such as those covered by the ecore formalism.

3.2 Transformation Rules

The previous B specifications specify only the structural aspects of source and target meta-models. The operational part is used to specify the transformation rules presented in section 2. In this section we present how transformation rules `translateGraph`, `translateNode` and `translateEdge` can be specified in B. Note that our objective is not to present an automatic translation from ATL into B, but to show, on the one hand, the shortcomings of MDE tools like ATL for a rigorous reasoning on transformations, and on the other hand, to present concretely the contribution of tools assisting a formal method. The following B operations are a way to specify the transformation from simple graphs to multi-graphs.

² EMF: Eclipsed Modeling Framework.

The first step of the transformation is to produce instances of class *MGraph* from instances of class *Graph*. This can be written in B using a simple operation in which set *Graph* becomes equal to set *MGraph*. In ATL there is no way to express this equality because source and target meta-models specify different worlds. Hence, OCL constraints that link these meta-classes is not possible.

```
generateMGraph =
BEGIN
  MGraph := Graph
END ;
```

The ATL rule **translateNode** creates instances of class *MNode* and instantiates their attributes *name* and *root*. It also updates links between instances of classes *MNode* and *MGraph*. The associated B operation follows the same principles and takes into account the total function *MGraph_Nodes* which assumes that an *MNode* must be associated to an *MGraph*.

```
generateMNode =
BEGIN
  MNode := Node ||
  MNode_name := Node_name ||
  MNode_root := Node_root ||
  MGraph_nodes := Graph_nodes
END;
```

Note that in ATL there is an explicit call of a transformation rule in rule **translateNode**. In B specifications operation call is not allowed in a same machine. Then, we dissociate these two rules in the specifications.

The ATL rule **translateEdge** creates edges from both roles *successor* and *predecessor* using lazy rule **translateEdge**. The associated B operation **generateMEdge** is similar and can be called on any couple $(n1, n2)$ such that $(n1, n2) \in predecessor_successor$. This operation introduces an *MEdge aa* in the model with $leaving_source(aa) = n1$ and $entering_target(aa) = n2$.

```
generateMEdge(n1, n2) =
PRE
  n1 ∈ Node ∧ n2 ∈ Node
  ∧  $(n1, n2) \in predecessor\_successor$ 
THEN
  ANY aa WHERE
    aa ∈ MEDGE - MEdge
THEN
  MEdge := MEdge ∪ { aa }
  || leaving_source(aa) := n1
  || entering_target(aa) := n2
END
END
```

4 Formal V&V Activities

This section shows what kinds of reasoning can be performed on the resulting B specifications. Having a formal specification of meta-models and transformation rules, two kinds of tools can be used: animators and provers.

4.1 Using an Animator

An animator like ProB [10] allows to simulate the transformation rules in order to test them. For example, the input model of figure 2 can be represented with valuations of constants of machine **SimpleGraphs**:

```
Graph = {GG}
^ Node = {N1, N2}
^ Graph_nodes = {(N1 ↦ GG), (N2 ↦ GG) }
^ Node_name = {(N1 ↦ AA), (N2 ↦ BB)}
^ Node_root = {(N1 ↦ TRUE), (N2 ↦ FALSE)}
^ predecessor_successor = {(N1 ↦ N2)}
```

Having these valuations, and following the ATL steps for this simple example, one can use ProB to animate the sequence of operations: `generateMGraph`, `generateMNode`, `generateMEdge(N1,N2)`, `generateMEdge(N1,N2)`.

Operation `generateMEdge(N1,N2)` is called twice because the ATL transformation of section 2 invokes twice rule `translateEdge`. The animation of this sequence of operations by the ProB tool produces the following state which is conform to the output model of figure 2:

```
MGraph={GG}
MNode={N1,N2}
MEdge={e1,e2}
leaving_source={(e1|->N1),(e2|->N1)}
entering_target={(e1|->N2),(e2|->N2)}
MNode_name={(N1|->AA),(N2|->BB)}
MNode_root={(N1|->TRUE),(N2|->FALSE)}
MGraph_nodes={(N1|->GG),(N2|->GG)}
```

The use of ProB allows to do at least what can be done by an MDE transformation tool (such as ATL). This is a first concrete advantage behind the use of a formal method assisted by such tools. ProB is also a model-checker which can be used to explore a set of possible initial models and also to try several executions of B operations. The animation generates, then, the target models from a set of source models and hence it allows to cover the a posteriori validation.

4.2 Using a Prover

In this section, we show the great interest of a prover based on our simple example. Our main objective is to introduce properties in the previous B specifications and to use the `atelierB` prover in order to amend the various ATL rules.

Model invariants. Firstly, we add an invariant to impose that there is one and only one root node in a multi-graph. This constraint was formalized in OCL in section 2 and it can be expressed in B by:

$$MNode \neq \emptyset \Rightarrow (\text{card}(MNode_root} \triangleright \{\text{TRUE}\}) = 1$$

By adding this invariant, the atelierB prover fails to prove associated proof obligations for operation `generateMNode`. Indeed this operation modifies, without any control, relation `MNode_root`. In order to remedy this failure, one can add the following precondition to the operation:

$$\text{card}(Node_root} \triangleright \{\text{TRUE}\}) = 1$$

This means that translation of nodes won't be possible for an input graph in which there are several root nodes. Consequently we can correct the ATL rule `translateNode` by adding an OCL pre-condition:

```
(SimpleGraph!Node.allInstances()->select(root = true)->size()=1)
```

There are several other ways to correct the specifications:

- transform one root node in accordance with the previous transformation and for the others turn their attribute root to false
- transform only one root node
- add an invariant to the source model in order to impose the existence of one and only one root node in a simple graph:

$$Node \neq \emptyset \Rightarrow (\text{card}(Node_root} \triangleright \{\text{TRUE}\}) = 1$$

This last invariant impacts the correctness of the source meta-model by adding the following OCL constraint:

```
context MGraph inv oneRootInputModel :
self.nodes->select(root=true)->size()=1
```

Meta-model links. Transformation properties which link both meta-models (stated previously and which are covered by existing MDE platforms) should be respected by the attainable state of the B operations. This is addressed by adding to machine `MultiGraphs` three boolean variables initialized to false (`graphDone`, `nodeDone` and `edgeDone`) and which are turned into true when a linkage property is satisfied. For example, the invariant which links meta-classes `Graph` and `MGraph` indicates that the set of produced instances of class `MGraph` is equal to the set of instances of source class `Graph`:

$$GraphDone = \text{TRUE} \Rightarrow Graph = MGraph$$

Operation `generateMGraph` which performs the transformation from `Graph` to `MGraph` classes is then as follows:

```
generateMGraph =
PRE graphDone = FALSE THEN
  MGraph := Graph ||
  graphDone := TRUE
END ;
```

Meta-models linkage property for node transformation (operation `generateMNode`) indicates that set `MNode` must be equal to set `Node`. Consequently, operation `generateMNode` becomes:

```
generateMNode =
PRE
  card(Node_root > {TRUE}) = 1
  ^ nodeDone = FALSE
THEN
  MNode := Node ||
  MNode_name := Node_name ||
  MNode_root := Node_root ||
  MGraph_nodes := Graph_nodes ||
  nodeDone := TRUE
END;
```

The invariant which links `Graph`, `MGraph`, `Node` and `MNode` is:

$$MGraph \subseteq Graph \wedge MNode \subseteq Node.$$

Finally, linkage invariant for relation `predecessor-successor` and variable `MEdge` constrains the number of produced edges. Indeed, the number of instances of class `MEdge` should be equal to the number of instances of association successor/predecessor. This leads to the following invariant:

$$\text{edgeDone} = \text{TRUE} \Rightarrow \text{card}(MEdge) = \text{card}(\text{predecessor-successor})$$

This possibility to link elements of source and target meta-models allows to complete the specifications by a more precise invariant which expresses the fact that classes being the source and the target of an `MEdge` are linked in the source model by association successor/predecessor:

$$\forall ee.(ee \in MEdge \Rightarrow (\text{leaving-source}(ee), \text{entering-target}(ee)) \in \text{predecessor-successor})$$

Operation `generateMEdge` presented previously doesn't respect this invariant. Such a violation is detected by the AtelierB prover and also by ProB when we animate our test case with several calls to `generateMEdge`. We then suggest the following correction in which transformation of edges is complete because it takes into account not only the linkage invariant but also the fact that a given couple of nodes ($n1, n2$) is considered once. Indeed, precondition

$$(n1, n2) \notin (\text{leaving-source}^{-1}; \text{entering-target})$$

avoids the transformation of the link predecessor/successor between $n1$ and $n2$ if there exists an edge between $n1$ and $n2$ in the resulting multi-graph.

```

generateMEdge =
PRE
  edgeDone = FALSE
THEN
  IF card(MEdge) < card(predecessor-successor) THEN
    ANY aa, n1, n2 WHERE
      aa ∈ MEDGE - MEdge
       $\wedge n1 \in Node \wedge n2 \in Node$ 
       $\wedge (n1, n2) \in predecessor\_successor$ 
       $\wedge (n1, n2) \notin (leaving\_source^{-1}; entering\_target)$ 
  THEN
    MEdge := MEdge ∪ {aa}
    || leaving_source(aa) := n1
    || entering_target(aa) := n2
  END
  ELSE
    edgeDone := TRUE
  END
END

```

The ATL transformation of section 2 produces edges twice from each couple of nodes linked by association successor/predecessor. The above discussion directs us to correct informally the ATL rule `translateNode` as follows:

```

rule translateNode {
  from
    nodeIn : SimpleGraph!Node
    (SimpleGraph!Node.allInstances()->select(root = true)->size()=1)
  to
    nodeOut : MultiGraph!MNode (
      name <- nodeIn.name,
      root <- nodeIn.root,
      leaving <- nodeIn.successor -> collect(ssSucc
        | translateEdge(nodeIn, ssSucc)),
      entering <- MultiGraph!MEdge.allInstances()
        ->select(s | s.target = nodeOut) -> asSet()
    )
}

```

In this ATL transformation instances of class *MEdge* are created when association *leaving* is formed for a given node *nodeOut*. Association *entering* is hence an update of links between instances of *MEdge* and *MNode*.

Rules ordering. In order to manage rules ordering, we add invariants like:

$$\begin{aligned} \textit{nodeDone} = \textbf{TRUE} \Rightarrow \textit{graphDone} = \textbf{TRUE} \wedge \\ \textit{edgeDone} = \textbf{TRUE} \Rightarrow \textit{nodeDone} = \textbf{TRUE} \end{aligned}$$

Obviously, for the prover these invariants need evolutions of the B specifications by adding preconditions to the operations. Thus, we add precondition `graphDone = TRUE` to operation `translateNodes` and `nodeDone = TRUE` to operation `translateEdges`. These evolutions will guide an animator for the choice of operations which the analyst can animate at each stage of the transformation.

5 Conclusion and Future Work

A great number of research works in MDE are dedicated to languages and techniques for the implementation of model transformations, but among these studies, few ones are interested by validation and verification problems. Testing techniques and proofs in their duality, are necessary in this context. This paper presented how a formal method such as B can be used in an MDE process in order to prove the correctness of manipulated models and their transformations.

The use of B for a specific model transformation was suggested in [7] by K. Lano. The author presented on the base of the well known UML-to-RDBMS example how the B formal method can be used to verify semantic properties of UML graphical models, and the correctness of transformations on these models. In this paper we try to generalize these concepts and show how other kinds of models which are simply specified by meta-models can be taken into account in a proof-based MDE approach. The formalisation proposed by [7, 8] is complex since it specifies transformation rules independently of their implementation technologies. In this paper we start from some ATL rules, and show how they can be specified in B in order to correct them. In our ongoing work we try to contribute towards an automatic generation of B operations from the ATL language. Furthermore, in the existing works formalization of meta-classes is done in separate machines which needs to take into account several dependencies in the specification of transformation rules. Our approach produces one B machine for every meta-model. We believe that this formalization is more natural since the transformation rules are directly encoded in the B machine of the target meta-model.

Contrary to existing approaches which tried to formalize model transformation, we proposed to adapt UML-to-B approaches in order to produce B specifications from source and target meta-models. We also showed how transformation rules can be encoded using B operations and what kinds of reasoning can be performed on the resulting B specifications.

Currently, we are developing a tool based on Eclipse Modeling Framework and which systematically produces B specifications from ecore meta-models. The systematic translation from transformation rules into B operations needs solid theoretical foundations. We believe that such perspective must be guided by the QVT standard of the OMG in order to be generalizable.

The possibility to use a model checker is not discussed in this paper, but it is an interesting benefit of our work. ProB, for example, can combine several possible executions of rules in order to try to find a possible transformation

which leads to an invariant violation. In the same direction, we plan to conduct studies on the automatic generation of input models. Indeed, if formal proofs are complex, the identification of relevant input models can help validate the transformation rules. Tools like BZ-TT [9] can be used to generate these models automatically from B specifications.

References

1. Abrial, J.-R.: *The B-book: assigning programs to meanings*. Cambridge University Press (1996)
2. Brottier, E., Fleurey, F., Steel, J., Baudry, B., Le Traon, Y.: Metamodel-based test generation for model transformations: an algorithm and a tool. In: ISSRE, pp. 85–94. IEEE Computer Society (2006)
3. Jouault, F., Kurtev, I.: Transforming models with atl. In: Bruel, J.-M. (ed.) MoDELS 2005. LNCS, vol. 3844, pp. 128–138. Springer, Heidelberg (2006)
4. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: Atl: A model transformation tool. *Sci. Comput. Program.* 72(1-2), 31–39 (2008)
5. Kurtev, I.: State of the art of qvt: A model transformation language standard. In: Schürr, A., Nagl, M., Zündorf, A. (eds.) AGTIVE 2007. LNCS, vol. 5088, pp. 377–393. Springer, Heidelberg (2008)
6. Laleau, R., Polack, F.: Coming and going from UML to B: A proposal to support traceability in rigorous is development. In: Bert, D., Bowen, J.P., Henson, M.C., Robinson, K. (eds.) ZB 2002. LNCS, vol. 2272, pp. 517–534. Springer, Heidelberg (2002)
7. Lano, K.: Using B to verify UML Transformations. In: MoDeVa: Model Development, Validation and Verification (October 2006)
8. Ledang, H., Dubois, H.: Proving model transformations. In: 2010 4th IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE), pp. 35–44. IEEE (2010)
9. Legeard, B., Peureux, F., Utting, M.: Automated boundary testing from Z and B. In: Eriksson, L.-H., Lindsay, P.A. (eds.) FME 2002. LNCS, vol. 2391, pp. 21–40. Springer, Heidelberg (2002)
10. Leuschel, M., Butler, M.: ProB: A Model Checker for B. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 855–874. Springer, Heidelberg (2003)
11. Mammar, A., Laleau, R.: From a B formal specification to an executable code: application to the relational database domain. *Journal of Information and Software Technology* 48(4), 253–279 (2005)
12. Ossami, D.D.O., Jacquot, J.-P., Souquières, J.: Consistency in UML and B Multi-view Specifications. In: Romijn, J.M.T., Smith, G.P., van de Pol, J. (eds.) IFM 2005. LNCS, vol. 3771, pp. 386–405. Springer, Heidelberg (2005)
13. Snook, C., Butler, M.: UML-B: Formal modeling and design aided by UML. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 15(1), 92–122 (2006)
14. Weisemöller, I., Schürr, A.: Formal definition of MOF 2.0 metamodel components and composition. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 386–400. Springer, Heidelberg (2008)

Managing Run-Time Variability in Robotics Software by Modeling Functional and Non-functional Behavior

Alex Lotz¹, Juan F. Inglés-Romero², Cristina Vicente-Chicote³,
and Christian Schlegel¹

¹ University of Applied Sciences Ulm, Germany
`{lotz,schlegel}@hs-ulm.de`

² Universidad Politécnica de Cartagena, Spain
`juanfran.ingles@upct.es`

³ QSEG, Universidad de Extremadura, Spain
`cristinav@unex.es`

Abstract. Service robots act in open-ended and natural environments. Therefore, due to the huge number of potential situations and contingencies, it is necessary to provide a mechanism to express dynamic variability at design-time that can be efficiently resolved on the robot at run-time based on the then available information. In this paper, we present a modeling process to separately specify at design-time two different kinds of dynamic variability: (i) variability related to the robot operation, and (ii) variability associated with QoS. The former provides robustness to contingencies, maintaining a high success rate in robot task fulfillment. The latter focuses on the quality of the robot execution (defined in terms of non-functional properties like safety or task efficiency) under changing situations and limited resources. We also discuss different alternatives for the run-time integration of the two variability management mechanisms, and show real-world robotic examples to illustrate them.

Keywords: Variability Management, Modeling Run-Time Variability, Service Robotics, SMARTTCL, VML.

1 Introduction

Service robots (e.g. companion, elder care, home health care, or co-worker robots) are expected to robustly and efficiently fulfill different tasks in complex environments (such as domestic, outdoor or crowded public spaces). Real-world environments are inherently open-ended and show a huge number of variants and contingencies. Thus, service robots need to know how to flexibly plan their sequence of actions in order to fulfill their tasks, taking into account changes in the environment, noisy perception and execution failures. For instance, a robot navigating in a building needs to reactively avoid dynamic obstacles in its local surrounding. Besides, it might need to reconsider its plan of how to get to its destination, e.g., in case the planned route is blocked. The management of

this *variability in operation* provides the robot with a high degree of robustness and allows it maintaining a high success rate in task fulfillment. On the other hand, there is typically a wide range of possibilities to succeed in the same task. Among these possibilities, some might be better than others according to quality criteria defined by the designer (e.g., in terms of resource consumption, safety, performance, etc.). For example, since robots are equipped with limited resources, they need to know how to spend them in the most appropriate way. Thus, if a robot is running out of battery, it might be a good idea to prioritize power consumption over task efficiency. The management of this *variability in quality* improves the overall robot execution performance.

Addressing variability in robotic software is nothing new. However, it has been traditionally managed in an ad-hoc way, i.e., developers try to predict future execution conditions and implement specific mechanisms to deal with each particular situation, spreading the variability management rationale throughout their application code. Among other issues, this usually leads to increased complexity, poor reuse, and it hinders the extensibility and maintenance of the applications. This motivates a different approach [1]: (i) we need to make it as simple as possible for the designer to express variability at design-time, and (ii) we need the robot to be able to bind variability at run-time, based on the then available information. At design-time, we propose to use two different Domain Specific Languages (DSLs), each one for modeling one of the previously described variability kinds: SMARTTCL [2] for *variability in operation*, and VML [1] for *variability in quality*. This way we encourage the separation of the two concerns: one for modeling *how* to coordinate the actions (e.g., a flexible plan considering contingencies), and another one for modeling *what is a good way* of achieving a task (e.g., in terms of non-functional properties like safety or power consumption). At run-time, we separate the variability management in two orthogonal mechanisms: (i) sequencing the robot's actions to handle *variability in operation*, and (ii) optimizing the non-functional properties for *variability in quality*. These two mechanisms enable the robot to decide on proper behavior variations by applying the design-time model information and taking the current situation into account. This approach improves the robustness and the task execution quality, optimizes robot performance and cleverly arranges complexity and efforts between design-time and run-time.

In this paper, we present a modeling process to separately specify *variability in operation* using SMARTTCL and *variability in quality* using VML. Besides, as one of the core contributions of this paper compared to our previous work on this topic [1], we analyze three different alternatives for a run-time integration of the two proposed variability management mechanisms in a safe and consistent way. We will describe the lessons learned and discuss the benefits and drawbacks of each alternative. In order to underpin the feasibility and the benefits of the proposal we provide a real-world case study in a robotics scenario. Despite the fact that our proposal is inspired by the robotic domain, we believe that the basic ideas we present in this paper are interesting and general enough to be considered in other domains.

2 Real-World Robotics Scenario

In order to demonstrate the variability modeling capabilities of the two individual languages, SMARTTCL and VML, we use the *Butler*¹ scenario. In this scenario we have a robot serving people. It carries out typical butler activities like taking orders, fetching beverages, cleaning up after customers left the place, etc. Although we have repetitive tasks in the scenario, the overall sequence of actions is never the same. Instead, the service robot must adjust its behavior according to changes in the environment, human orders, etc.

To achieve this flexibility in the robot operation, SMARTTCL allows designers to model possible contingencies of the scenario. For instance, in the *clean up the table* activity, the robot has to take objects like glasses and cups to the kitchen, and place others, like tetra packs or empty cans, in a trash bin. Then, what happens if the robot finds unexpected objects on the table? Although the robot can presume which objects are left on a table from the previous orders, the real situation might be quite different, as customers might have left their glasses on different tables or forgotten personal objects such as a mobile phone. Besides, typically, the sequence of appropriate actions for the clean up task strongly depends on each particular situation, and needs to take into account both robot limitations and constraints (e.g., its physical capacity to carry only a limited weight or the maximum number of cups it can stack into each other to safely carry them to the kitchen), and application-specific information (e.g., which objects must be thrown away and which ones must be cleaned and reused). Thus, the most robust way for a service robot to operate is to react on each situation with as little assumptions as possible. This motivates an approach where the behavior of the robot must be as flexible as possible, just defining the strategies for how to react on situations by finding appropriate sequences of actions to achieve, e.g., the clean up task.

Now, imagine that we want to optimize the *coffee delivery*² service. The robot has to trade-off various aspects to come up with an improved quality of service. Thus, it needs to be able to select an appropriate velocity to properly fulfill its task according to further issues like safety or energy consumption: (i) customers are satisfied only if the coffee has at least a certain temperature, but prefer it as hot as possible; thus, serving fast is relevant, (ii) however, the maximum allowed velocity is bound due to safety issues (hot coffee) and also by battery level, (iii) since coffee cools down depending on the time travelled, a minimum required average velocity (depending on distance to customer) is needed, although driving slowly makes sense in order to save energy, (iv) nevertheless, fast delivery can increase the volume of sales. Although the main functionality of the robot is to deliver coffee, regardless of how well it is performed, VML can help designers to model, on top of this functionality, QoS policies based on (often conflicting) non-functional properties. As a result, the variability (e.g., robot maximum allowed

¹ Butler scenario video: <http://www.youtube.com/user/RoboticsAtHsUlm>

² Coffee scenario video: <http://www.youtube.com/watch?v=-nmlixl9kik>

velocity) is bound at run-time to optimize these policies according to the current application context (e.g., the battery level or how crowded the coffee shop is).

Inspired on the above example, we consider that the Butler scenario takes place in a room with two coffee machines located in different positions. When someone asks the robot for a cup of coffee it must decide: (i) which coffee machine to use, and (ii) its maximum allowed velocity. This decision is made at run-time in order to improve the quality of the service taking into account power consumption (e.g., when the battery is low the system must optimize power consumption using the nearest coffee machine) and performance (e.g., trying to get the highest value for maximum allowed velocity in order to reach the goal faster). Obviously, maximizing performance while simultaneously minimizing power consumption imposes conflicting requirements. To deal with this, VML allows expressing, at design-time, the existing dependencies among conflicting requirements such that, at run-time, the robot can find the right balance among them.

3 Modeling Variability with SmartTCL

We use the TASK COORDINATION LANGUAGE (SMARTTCL [2]) to model *variability in operation*. The main purpose of SMARTTCL is to define rules and strategies that specify *how* the system behaves when accomplishing different tasks. SMARTTCL allows to react to changes in the environment and to adjust task execution according to the current situation.

3.1 SmartTCL Syntax

The SMARTTCL EBNF grammar is defined in Listing 1.1. The main element of SmartTCL is the *Task Coordination Block* (TCB). A TCB represents an abstract task (e.g., moving to a generic location), and its function is to *orchestrate* (configure and activate) the system components to execute the proper primitive actions needed to achieve this task. A TCB is defined by its signature, consisting of the name and the in/out parameters and, optionally, a *precondition* clause and a *priority* that, when available, help selecting the most appropriate TCB at run-time. The *body* of a TCB contains, at least, an *action* block, a *plan* or both. An *action* block is used to define primitive behaviors encoded in Lisp. A *plan* is used to establish a parent/child relationship between TCBs and, thus, to create complex behaviors. This enables SMARTTCL to define recursion (with the termination clause encoded as the precondition) and to create *task-trees* consisting of TCBs as nodes. The plan is not static, but can be dynamically adjusted at run-time, e.g., by asking a symbolic planner (which allows to generate action plans at run-time) for a sequence of TCBs. The default execution semantics for the TCBs in a *plan* is to execute them one after the other in sequence. Alternatively, it is possible to execute all the TCBs in a *plan* in *parallel* (waiting until they have all finished), or use the *one-of* semantics (again in parallel but, this time, exiting as soon as one of the TCBs finishes). In addition, the *body* of a

```

SmartTCL ::= ( TCB | [EventHandler] | [RuleDef] )+
(* TCB definition *)
TCB ::= '(' Signature Body ')'
TCB_NAME ::= ID
LISP_CODE ::= STRING
(* TCB-Signature definition *)
Signature ::= 'define-tcb' '(' TCB_NAME InParams '>' OutParams ')'
[Precondition] [Priority]
InParams ::= ('?' ID)*
OutParams ::= ('?' ID)*
Precondition ::= '(' 'precondition' '(' LISP_CODE ')') '
Priority ::= '(' 'priority' INT ')'
(* TCB-Body definition *)
Body ::= [Rules] (ActionBlock [Plan] | [ActionBlock] Plan) [AbortActions]
Rules ::= '(' 'rules' '(' (ID)+ ')') '
ActionBlock ::= '(' 'action' '(' LISP_CODE ')') '
AbortActions ::= '(' 'abort-action' '(' LISP_CODE ')') '
Plan ::= '(' 'plan' '(' ['parallel'|'one-of'] (TCB_NAME)+ ')') '
(* EventHandler definition *)
EventHandler ::= '(' 'define-event-handler' '(' ID ')') ActionBlock ')
(* Rule block definition *)
RuleDef ::= '(' 'define-rule' '(' TcbBinding TcbEvent ActionBlock ')') '
TcbBinding ::= '(' 'tcb' '(' TCB_NAME InParams '>' OutParams ')') '
TcbEvent ::= '(' 'tcb-return-value' '(' TcbEventCode('TcbEventDescription'))')') '
TcbEventCode ::= ('SUCCESS' | 'ERROR')
TcbEventDescription ::= (STRING)*

```

Listing 1.1. EBNF grammar for SmartTCL

TCB can optionally contain a sequence of *rules* (see below) or a block of *abort-actions*. The latter implement cleanup procedures in case the TCB is aborted before completion.

In addition to TCBs, SMARTTCL defines *EventHandlers* and *Rules*. *EventHandlers* are used to receive feedback from the components in the system. This feedback can signal, e.g., successful completion or a failure in the execution of a previously triggered action. In any case, the *EventHandler* executes a block of actions to appropriately react on that event. A *rule* is a very handy mechanism to react on contingencies during the execution of the TCBs in a plan. This mechanism is comparable with C++ Exceptions. If one of the child TCBs has a problem during its execution that cannot be solved locally, it can propagate an error message up the hierarchy, which is then caught by the first fitting *rule*. For each TCB, various rules can be defined (indicated by the binding part of the rule and associated to a certain event) to provide different strategies to react on contingencies. A parent TCB activates a set of rules for its child TCBs in order to create appropriate recovery strategies. This considerably improves the reuse and flexibility of the TCBs in different scenarios.

3.2 SmartTCL Execution Semantics

At run-time, on the robot, SMARTTCL plays the coordinator role by orchestrating software components. Thereby, each TCB represents a certain, consistent system state with all the configuration parameters for individual components.

The main functionality of SMARTTCL is illustrated by the *clean-up table* scenario introduced in Sec. 2. The whole scenario is modelled on the robot using

SMARTTCL. There, a user operating with the robot can command it to clean up the dinner table, which activates the TCB labelled in Listing 1.2 and in Fig. 1 as ①). This TCB has an input parameter named ?location. From the parent TCB this parameter is set to dinner-table, which is a placeholder that can be resolved in the knowledge base (like in ③). It is worth noting that, in our case, a knowledge base is simply a database (a memory) to store key/value pairs (for e.g. the robot model, the TCBs, environment model, etc.).

```
(define-tcb (cleanup-table ?location)
  (rules (rule-action-cleanup-table-failed rule-action-cleanup-table-empty
    rule-action-cleanup-say-success ))
  (plan (
    (tcb-approach-location ?location)
    (tcb-say "I have reached the table, and will look for objects to clean up.")
    (tcb-cleanup-table))))
```

Listing 1.2. Cleanup table TCB definition

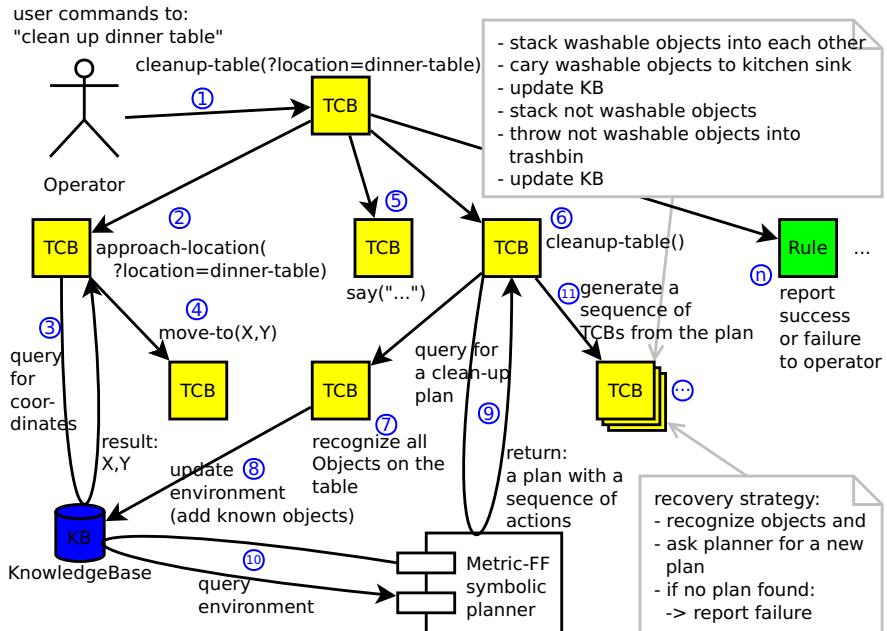


Fig. 1. Snapshot of a task-tree at run-time from the clean-up scenario example

As shown in Listing 1.2, the *cleanup-table* TCB has no *action* block, but it defines a *plan* comprising other TCBs to approach a location ② (in this case the dinner-table), to announce via speech that the robot is now going to look for objects to clean up ⑤, and then to execute the TCB *cleanup-table* ⑥, which is different from its parent because of its different signature. In addition, a list of *rules* is activated which, in this case, altogether belongs to the child

TCB `cleanup-table`. These rules define the behavior of this TCB according to certain results in the execution. For instance, if something goes wrong while cleaning up the table, the first rule can trigger a re-planning or can delete the current plan and report the failure to the operator. In case the cleaning up succeeds, the robot can update its internal environment representation and report the success to the operator (see also (n) in Fig. 1).

It is important to notice that task-trees are created and evolve (according to changes in the environment, execution failures, etc.) only at run-time. It is also noticeable that, in contrast to regular finite state machines, we do not model the transitions between the TCBs. Instead we define rules and strategies that specify how a TCB can be expanded and refined at run-time taking the then available information into account. This leads to a much more flexible and robust system behavior. It is even possible and a regular case to generate complete (sub)trees by asking a symbolic planner (see (9) and (11)). In this example, the planner returns a sequence like: stack cup A into cup C and then cup C into cup B, etc. This is a powerful mechanism we call *expert usage* that will be handy for interacting with VML (see Sec. 5). In summary, we use *experts*, *rules* and *event handlers* to design *variability in operation*.

4 Modeling Variability with VML

In this section we introduce the *Variability Modeling Language* (VML) that provides a mechanism to express *variability in quality*, that is, how a system should adapt at run-time to maintain or improve the system execution quality under changing conditions. The current version of VML has been developed using a *Model-Driven Engineering* (MDE) approach. We have created a textual editor for VML using the Xtext framework³, including some advanced features such as syntax checking, coloring and a completion assistant.

In a VML model, we first need to define the variation points. Aligned with *Dynamic Software Product Lines* (DSPL) [3], VML variation points represent points in the software where different variants might be chosen to derive the system configuration at run-time. Therefore, variation points determine the decision space of VML, i.e., the answer to *what* can change. As shown in Listing 1.3, variation points (`varpoint`), as all the other VML variables, belong to a certain data type. VML includes three basic data types: enumerators, ranges of numbers and booleans. For instance, `maximumVelocity` is a variation point to limit the maximum velocity of the robot, which takes values from 100 to 600 with precision 10 mm/s. Similarly, the `coffeeMachine` variation point is an enumerator that gathers the two available coffee machines that can be used by the robot: `COFFEE_MACHINE_A` and `COFFEE_MACHINE_B`. Once we have identified the variation points, context variables (`context`) are used to express situations in which variation points need to be adapted. Listing 1.3 shows five context variables: (i) the battery level (integer value in the range 5-100), (ii) distance to

³ Xtext: www.eclipse.org/Xtext

each coffee machine (real number in the range 0-20 with precision 0.1), and (iii) waiting time at each coffee machine (integer in the range 10-300), taking into account the operation time of the machine (constant time) and the time the robot has to wait because there are other users (robots or people) waiting for it (variable time).

```
/* Contexts variables */
context ctx_battery : number [0:1:100];
context ctx_distanceMachine_A : number [0:0.1:20];
context ctx_distanceMachine_B : number [0:0.1:20];
context ctx_waitingTimeMachine_A : number [10:1:300];
context ctx_waitingTimeMachine_B : number [10:1:300];

/* Auxiliary variables */
var timeMachine_A := ctx_waitingTimeMachine_A + ctx_distanceMachine_A/600;
var timeMachine_B := ctx_waitingTimeMachine_B + ctx_distanceMachine_B/600;

/* ECA rules */
rule lowBattery_NearMachineA :
  ctx_battery < 15 and ctx_distanceMachine_A < ctx_distanceMachine_B
  => coffeeMachine = @coffeeMachine.COFFEE_MACHINE_A;
rule lowBattery_NearMachineB :
  ctx_battery < 15 and ctx_distanceMachine_A >= ctx_distanceMachine_B
  => coffeeMachine = @coffeeMachine.COFFEE_MACHINE_B;
rule high_EFF_coffeeMachA :
  ctx_battery >= 15 and timeMachine_A > timeMachine_B
  => coffeeMachine = @coffeeMachine.COFFEE_MACHINE_A;
rule high_EFF_coffeeMachB :
  ctx_battery >= 15 and timeMachine_A <= timeMachine_B
  => coffeeMachine = @coffeeMachine.COFFEE_MACHINE_B;

/* Properties to optimize */
property performance : number[0:1:100] maximized {
  priorities:
    f(ctx_battery) = max(exp(-1*ctx_battery/15), 10 sec) - exp(-1*ctx_battery/15);
  definitions:
    f(maximumVelocity) = maximumVelocity;
}
property powerConsumption : number[0:1:100] minimized {
  priorities:
    f(ctx_battery) = exp(-1 * ctx_battery/15);
  definitions:
    f(maximumVelocity) = exp(maximumVelocity/150);
}

/* Variation points */
varpoint maximumVelocity : number [100:10:600];
varpoint coffeeMachine : enum { COFFEE_MACHINE_A, COFFEE_MACHINE_B };
```

Listing 1.3. VML Model for choosing Coffee Machine

At this point, we need to define how variation points are set according to the context. This is achieved through properties (*property*) and Event-Condition-Action (ECA) rules (*rule*). Properties specify the features of the system that need to be optimized, i.e., minimized or maximized. Properties are defined using two functions: *priorities* and *definitions*. Definitions characterize the property in terms of variation points (i.e., definitions are the objective functions to be optimized). For instance, in Listing 1.3, we define the performance property

as a linear function of the maximum velocity variation point (the faster the robot accomplishes its task, the better its performance). Similarly, the power consumption property also depends (in this case, exponentially) on the maximum velocity (the faster the robot moves, the higher its power consumption). It is worth noting that property *definitions* are characterized using the technical specifications of the hardware (e.g., to know how much power each component consumes), simulation or empirical data from experiments. On the other hand, *priorities* describe the importance of each property in terms of one or more context variables (i.e., priorities weight the objective functions). For instance, power consumption becomes more and more relevant as the robot battery decreases. In fact, when the battery is full, power consumption is not considered an issue and, as a consequence, the priority of this property in that context is very small. Opposite to definitions, priorities are characterized in a more subjective way, depending on the designer experience.

Regarding the ECA rules, they define direct relationships between context variables and variation points. As shown in Listing 1.3, the left-hand side of a rule expresses a trigger condition (depending on one or more context variables) and its right-hand side sets the variation point. For example, the decision of which coffee machine to select in each situation is modeled using rules. Basically, the first two rules are applied when the battery is low (less than 15%) to select the nearest coffee machine (reducing travel distance lowers power consumption when the battery is critical). Conversely, if the battery is high enough, the last two rules select the machine with lower waiting time (reducing the coffee delivery time improves performance when the battery is not an issue).

Regarding execution semantics, VML models specify a constrained optimization problem, that is, it describes the global weight function that optimizes the variation points to improve the overall system quality. This global function is obtained by aggregating the property definitions (terms to be optimized), weighted by their corresponding priorities. Besides, the ECA rules state constraints that need to be satisfied. Therefore, in order to execute a VML model, we transform it into *MiniZinc* [4] (a constraint programming language), so that the generated code is used as an input by the *G12 Constraint Programming Platform*⁴ to solve the constraint problem. At this point, it is worth noting that VML variation points and contexts are high-level variables that somehow abstract architectural elements (e.g., components or component parameters). For instance, the maximum velocity is linked to a parameter of the motor component, and the battery level is obtained from a sensor component. This abstraction allows VML to be independent of the underlying architecture, what, among other benefits, enables the reuse of the specification. However, when it comes the time to map the abstract VML elements to the actual system architecture, we must carefully take into account how this might interact with the decisions made by SMARTTCL. Next section explains how both variability management mechanisms have been finally integrated in a safe and consistent way, after assessing different approaches.

⁴ G12: www.g12.csse.unimelb.edu.au

5 Run-Time Integration of SmartTCL and VML

To this point, we have explained how to use SMARTTCL and VML to separately specify the *variability in operation* and the *variability in quality*, respectively. This section addresses one of the key issues when using both variability management mechanisms together, i.e., how to integrate them in a safe and consistent way. Next, we describe chronologically the integration approaches we followed to deal with this issue.

As detailed in Sec. 3, the role of SMARTTCL is to orchestrate the changes in the software architecture, i.e., to configure, activate and deactivate the components, according to an action plan, to enable the robot to fulfill its tasks. On the other hand, the VML engine sets a number of variation points (with impact in the architecture components or in some of their parameters), so that the overall QoS delivered by the robot is optimized. Our first integration approach was to enable both mechanisms to have impact on the software architecture. However, although we can advise the designer to create SmartTCL and VML models having orthogonal decision spaces, formally checking this orthogonality at design-time is a hard problem. We evaluated the possibility of implementing a tool that could help the designer in the creation of mutually consistent SmartTCL and VML models (i.e., guaranteeing that both models would not produce conflicting decisions at runtime). However, as the number of potential situations the system could have to face is unbounded and first known at run-time, the potential decisions of both mechanisms in response to those situations are also unbounded. Thus, we discarded this initial approach as it was not feasible in practice.

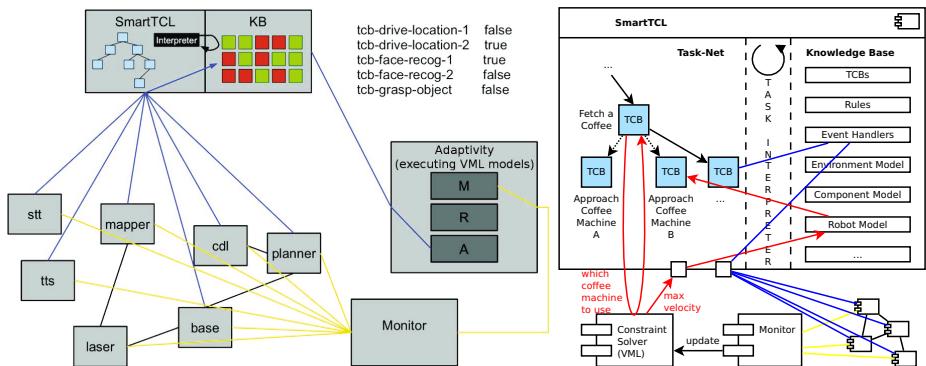


Fig. 2. On the left: discrete TCBs to handle variants during optimization. On the right: expert usage and continuous updates

In the robotics domain, one general approach to face decision conflicts is to define a single master in the system, which is responsible for making the final decisions at run-time. In our case, there is a natural hierarchy already available. Thereby, SMARTTCL models all functional aspects related to the *variability in operation*. In designing such models, typically one finds several solutions with

similar success expectations regarding the goals of the tasks. From now on, we will call these different solutions for the same task *variants*. These variants can be used as an input for the VML engine which, at run-time, will select one out of them (the best possible one) by optimizing the non-functional properties defined in the VML model. This way, the VML engine will advise SMARTTCL on its decisions but will not interfere with the overall system functionality. As shown in Fig. 2 (on the left) we have implemented such *variants* as extra TCBs, which have an additional boolean flag indicating whether the TCB is activated (green boxes) or not (red boxes). At run-time, the VML engine (executed by the Adaptivity component): (i) monitors (M) the system to determine the current situation, (ii) then reasons (R) about the appropriate adjustments, and (iii) activates (A) one out of the alternative TCBs. SMARTTCL makes the final decision using only the TCBs that are active in that moment. This already allows to handle problems similar to the coffee delivery example, where one TCB would be for approaching coffee machine A and another TCB for coffee machine B. Although this approach works fine for simple scenarios, where discrete decisions must be taken, it is highly inefficient for continuous variation points. For example, the decision on the maximum allowed velocity would require: (i) deciding about the most appropriate discretization for the continuous variable, and (ii) creating a TCB for each discrete value. One could define a few discrete values like SLOW, MEDIUM and FAST, but the problem would remain unsolved when trying to assign concrete values to these labels in all potential situations. Another limitation of this approach is that all the variability must be statically defined at design-time (as concrete TCBs) and cannot be adjusted anymore at run-time according to different situations.

The limitations of the previous method motivated the need for a different approach that could directly deal with numeric and even floating point values. This approach follows the *subsidiarity principle*. Similar to a company where budgets are assigned down the hierarchy, and the lower departments manage their limited budgets without conflicting with the overall company decisions, a similar approach is applied in our software architecture. Therefore, we define contexts and variation points as the two interfaces between SMARTTCL and VML. A *context* represents the current situation and the robot state. Contexts can be determined either directly from the knowledge base or deduced from an additional (monitoring) component, which infers certain information out of the raw data available in the system. A *variation point* is then a system variable (allowing either discrete or continuous values), whose boundaries are not static, but can be adjusted at run-time from SMARTTCL according to the current functional needs. The remaining variability is then used by VML to decide on concrete values (subsidiarity principle). At run-time there are two mechanisms that implement this behavior. As mentioned in Sec. 3, SMARTTCL allows to use different *experts* during execution (see Fig. 2 on the right). In our case, one of these experts is going to be the VML engine (constraint solver). For example, before expanding a TCB for fetching a coffee, SMARTTCL sends a query to the VML engine expert asking for advise on which is the best coffee machine

to go. SMARTTCL will attach the pertinent context information in the query so that the VML engine can make an informed decision. For the continuous variation points a different mechanism is used. For example, for the selection of the maximum allowed velocity, an additional variable in the knowledge base is used (the robot model in Fig. 2 on the right). Thereby, SMARTTCL adjusts this variable according to its functional needs, and VML is only allowed to set this variable within the remaining decision space. The VML engine is triggered whenever the monitor detects a situation that requires to adjust the maximum allowed velocity (e.g., when the environment is crowded). When this happens, the constraint solver of the VML engine calculates the optimum value for the variation point, and informs SMARTTCL which, in turn, updates the knowledge base. In summary, this approach enables SMARTTCL to be aware of VML models, and to ensure consistency (i) either by asking for advice about the possible alternatives, not conflicting with the overall task, or (ii) by propagating an already constrained variable to VML so that it can further constrain it without conflicting with the operational part.

6 Related Work

Service robotics is a challenging domain that exhibits a clear need for modeling and managing variability: robots should be able to change their behavior in response to changes in their operation or their environment. One step to solve this problem is to introduce mechanisms to model robot tasks independently of the components. In this sense, some work [5, 6] aims to rapidly compose complex robot behaviors based on state machines. These approaches support static composition of behaviors with little capabilities for situation dependent task adjustments. Therefore, designers need to include in the description of the functionality, which contingencies may occur for each situation, with almost no reuse. Conversely, SMARTTCL allows to easily model dynamic task trees, which are rearranged at run-time according to the current situation.

Some other works have been applied to robotics, although they are not focused on this domain. Among them, DiaSpec [7] is a DSL to specify Sense/ Compute/ Control (SCC) applications, where the interaction with the physical environment is essential. DiaSpec allows designing applications independently of the underlying architecture, describing entities (e.g., components) and controllers, which execute actions on entities when a context situation is reached. The DiaSpec compiler generates Java code that needs to be completed with the application logic. Like SMARTTCL and VML, DiaSpec platform-independence enables specification reuse. However, DiaSpec adaptation mechanisms completely rely on ECA-based rules and do not support any kind of optimization. Other works, like [8] and [9], have introduced the term architecture-based adaptation, applied to component-based robotic systems. In these works, components are replaced or migrated at run-time (e.g., due to a failure or resource insufficiency) by similar components with differently implemented services. The authors in [8] propose to model the variability through adaptation policies, included in the architecture definition. The basic building blocks of adaptation policies are observations

and responses. Observations establish information about a system and responses determine modifications in the architecture (additions and removals of architectural elements). The run-time management of policies is addressed by adopting an expert system approach. PLASMA [9] uses an *Architecture Definition Language* (ADL) and a planning-as-model-checking technology to enable dynamic re-planning in the architectural domain. That is, PLASMA uses ADL models and system goals as inputs for generating new plans in response to changes in the system requirements and goals, and in the case of unforeseeable component failures. In contrast to our approach, where we separately model and manage variability related to the robot operation and to the QoS, there is no such separation in any of the previous approaches. Similarly to SMARTTCL, PLASMA has the capability of dynamic re-planning. However, the approach presented in [8], as DiaSpec, only uses ECA-based rules.

Although the literature about dynamic variability is quite extensive, we would like to highlight some general-purpose frameworks, like Rainbow⁵ and MUSIC⁶, that enable self-adaptive system development. As the previous approaches, these frameworks are focused on component-based architectures. Rainbow proposes Stitch [10], a language for expressing adaptation strategies which, among other capabilities, allows representing QoS objectives. Quite similarly to Rainbow, MUSIC manages architecture reconfigurations via goal policies, expressed as utility functions. Each component implementation is associated with some predictors, which precisely specify the impact of a particular implementation on QoS properties. A global utility function computes the overall utility of the application to evaluate different configurations and choose the best one. The idea behind Rainbow and MUSIC is similar to VML but, as [8] and [9], they are too coupled to the underlying architecture. Moreover, they do not enable multiple levels for modeling and managing variability.

In the field of Dynamic Software Product Line (DSPL) [3], MOSKitt4SPL⁷ enables designers to model dynamic variability by means of (i) feature models, describing the possible configurations in which the system can evolve, and (ii) resolution models, defining the reconfigurations in terms of feature activation/deactivation associated with a context condition. This specification is automatically transformed into a state machine. Like VML and SmartTCL, this approach enables specification reuse at design-time. However, it does not support expressing optimization of QoS. Also in this line, the DiVA⁸ Project provides a tool-supported methodology with an integrated framework for managing dynamic variability. It is worth noting that VML has been inspired by the DiVA DSL [11], which allows managing variability using both ECA rules and optimization of QoS properties. However, DiVA relies on fuzzy logic to capture and describe how systems should adapt. Conversely, VML offers a more precise and less limited way to describe variability, e.g., using mathematical expressions and

⁵ Rainbow: www.cs.cmu.edu/~able/research/rainbow

⁶ MUSIC: <http://ist-music.berlios.de/site/index.html>

⁷ MOSKitt4SPL: <http://www.pros.upv.es/m4spl/features.html>

⁸ DiVA: <http://www.ict-diva.eu/>

real variables. This provides designers with a more natural way for describing the variability of their systems, in particular, in some application domains like in robotics. Finally, it is worth noting that both MOSKitt4SPL and DiVA only support *variability in quality*, but not *variability in operation*.

7 Conclusions and Future Work

In this paper we have presented two different DSLs, SMARTTCL and VML, which enable robotics software designers to separately specify *variability in operation* and *variability in quality*. Managing the former with SMARTTCL provides robustness to contingencies, maintaining a high success rate in task fulfillment. On the other hand, managing *variability in quality* with VML focuses on improving the overall execution performance of the robot under changing situations and limited resources. We have also discussed different possibilities for integrating both variability management mechanisms at run-time in a consistent way, i.e., avoiding conflicting reconfiguration decisions.

Regarding the run-time integration of SMARTTCL and VML, we highlight three main contributions: (i) we propose modeling variability at two different levels so that each one has its own (orthogonal) decision space, minimizing the potential conflicts between them (e.g., SMARTTCL configures components and VML relies on SMARTTCL), (ii) the VML engine, aimed to improve the overall system performance, is not essential and can be eventually stopped, i.e., the system must be fully operative without it, although performing suboptimally, and (iii) SMARTTCL and VML enable model reuse, i.e., the same specifications can be used for different platforms or applications (e.g., a TCB to move to a generic location or a VML definition of how to optimize the power consumption can applied in many scenarios).

For the future, we plan to use *Model Driven Engineering* to develop design-time tools for SMARTTCL. This would allow creating advanced editors to support developers in their software design and, in particular, to connect both SMARTTCL and VML on the meta-model level. This would further reduce the modeling efforts and would better support separate developer roles. We also plan to extend VML with some additional syntax constructs and to improve the supporting tools to provide designers with some advanced model validation and simulation facilities.

Acknowledgements. The project work at University of applied Sciences Ulm is found by ZAFH Servicerobotik⁹. The collaboration is jointly founded by the German Academic Exchange Service (DAAD; Project ID: 54365646; Project leader: Prof. Schlegel) and the General Directorate of International Projects of the Spanish Ministry of Economy and Competitiveness (MINECO; Project leader: Dr. Vicente-Chicote; Project ID: PRI-AIBDE-2011-1094). Juan F. Inglés-Romero thanks Fundación Séneca-CARM for a research grant (Exp. 15561/FPI/10).

⁹ <http://www.zafh-servicerobotik.de>

References

- [1] Inglés-Romero, J.F., Lotz, A., Vicente-Chicote, C., Schlegel, C.: Dealing with Run-Time Variability in Service Robotics: Towards a DSL for Non-Functional Properties. In: 3rd International Workshop on Domain-Specific Languages and models for ROBotic systems (DSLRob 2012), Tsukuba, Japan (November 2012)
- [2] Steck, A., Schlegel, C.: Managing execution variants in task coordination by exploiting design-time models at run-time. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, California, USA, pp. 2064–2069 (September 2009)
- [3] Hallstein sen, S., Hin chey, M., Park, S., Schmid, K.: Dynamic software product lines. Computer 41(4), 93–95 (2008)
- [4] Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.R.: MiniZinc: Towards a standard CP modelling language. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 529–543. Springer, Heidelberg (2007)
- [5] Bohren, J., Cousins, S.: The smach high-level executive (ros news). IEEE Robotics Automation Magazine 17(4), 18–20 (2010)
- [6] Dhouib, S., Kchir, S., Stinckwich, S., Ziadi, T., Ziane, M.: RobotML, a Domain-Specific Language to Design, Simulate and Deploy Robotic Applications. In: Noda, I., Ando, N., Brugali, D., Kuffner, J.J. (eds.) SIMPAR 2012. LNCS, vol. 7628, pp. 149–160. Springer, Heidelberg (2012)
- [7] Bertran, B., Bruneau, J., Cassou, D., Loriant, N., Balland, E., Consel, C.: DiaSuite: a Tool Suite To Develop Sense/Compute/Control Applications. Science of Computer Programming, Fourth special issue on Experimental Software and Toolkits (2012)
- [8] Georgas, J.C., Taylor, R.N.: Policy-based architectural adaptation management: Robotics domain case studies. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Self-Adaptive Systems. LNCS, vol. 5525, pp. 89–108. Springer, Heidelberg (2009)
- [9] Tajalli, H., Garcia, J., Edwards, G., Medvidovic, N.: Plasma: a plan-based layered architecture for software model-driven adaptation. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, ASE 2010, pp. 467–476. ACM (2010)
- [10] Cheng, S.-W., Garlan, D.: Stitch: A language for architecture-based self-adaptation. Journal of Systems and Software, Special Issue on State of the Art in Self-Adaptive Systems 85(12) (December 2012)
- [11] Fleurey, F., Solberg, A.: A domain specific modeling language supporting specification, simulation and execution of dynamic adaptive systems. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 606–621. Springer, Heidelberg (2009)

The World Out There: From Systems Modelling to Enterprise Modelling

Sobah Abbas Petersen¹ and John Krogstie²

¹ SINTEF Technology & Society, Norway
sobah.petersen@sintef.no

² Norwegian University of Science & Technology, Norway
krogstie@idi.ntnu.no

Abstract. In this paper, we describe how Masters students at a university studying Information Systems adapt to Enterprise Modelling; expressed metaphorically, they come out of their UML tunnels and see the world outside. The paper describes an overview of the course and the results and observations of the course based on the modelling assignments submitted by the students. The main observation is that while students are able to conceptualise a situation and represent that as an Enterprise Model, they often lack the experience to clearly state the purpose of the model and therefore, lack the ability to evaluate their models appropriately. Based on the experiences of teaching Enterprise Modelling to Information Systems students, the paper also provides a set of recommendations for teachers.

Keywords: Enterprise Modelling, Model Evaluation, Action Research, Modelling Practice, Aspects of modelling, Education.

1 Introduction

The goal of an enterprise model is to support analysis of an enterprise and to model the relevant business processes and how they relate to the rest of the enterprise such as how the work is organized and resource utilization, [1], [2]. The question "is Enterprise Modelling an art or science?" has often been asked and discussed; e.g. in a breakout session in the conference on The Practice of Enterprise Modelling, PoEM'10. The discussion if the area is an art or a science has sometimes surfaced in other domains such as management. In the context of management and evaluation of learning programs, they have been considered as a science which is organized knowledge – concepts, theory, principles and techniques. As an art, it is the application of the organised knowledge to concrete situations, usually with blend or compromise, to obtain desired results [3]. These descriptions of art and science seem to fit the domain of Enterprise Modelling as well; as a science; it is the formalisms, concepts, modelling methods, notations and languages. When this knowledge is applied to create Enterprise Models in practice, it becomes an art. However, to ensure that the knowledge is applied in an appropriate way to gain optimal results from the modelling, it is important to consider the *practice* of Enterprise Modelling.

In this paper, we describe how Masters students at a university studying Information Systems adapt to Enterprise Modelling; expressed metaphorically, they come out of their UML tunnels and see the world outside. This paper will present our experience from two years of teaching Enterprise Modelling. The aim of the course is to introduce Enterprise Modelling and to teach the students to create enterprise models and get them to practice Enterprise Modelling. The course was designed to reflect the different levels in the Bloom's taxonomy of learning goals [4], in particular, applying, analysing and evaluating. An Action Research [5] approach was taken by the teachers, where a cycle of planning, action and reflection was considered. The aim of the teachers was to improve the curriculum and teaching approach and practice by reflecting upon the previous years' courses and making improvements for the next year. The focus of the teachers was on how the students applied the theory that they have learned to create Enterprise Models and how they ensured that their models served their intended purposes. The main aim of this paper is to report our experiences as recommendations to other people that teach Enterprise Modelling.

The observations and conclusions are based on the modelling assignments delivered by the students. The main observation is that while students are able to conceptualise a situation and represent that as an Enterprise Model, they often lack the experience to clearly state the purpose of the model and therefore, lack the ability to evaluate their models appropriately. A model may be created to represent a variety of situations for a range of purposes, e.g. to achieve a common understanding among different people in the enterprise [2]. In addition, the aim of the modelling exercise may be to fulfill a very specific purpose such as help in a specific process within an enterprise or support decision making. As a part of evaluating the model, it is important to ensure that the model serves its purpose.

The rest of this paper is organized as follows: Section 2 provides a background of Enterprise Modelling and some related work; Section 3 describes the Enterprise Modelling course on which the paper is based upon; Section 4 provides an example of a model that was created by one of the students; Section 5 describes the observations from the modelling assignments of the course; Section 6 provides a set of recommendations for other teachers and Section 7 summarises the paper and reports on the future plans.

2 Enterprise Modelling

There are several definitions of Enterprise Modelling, from different perspectives; e.g. Vernadat defines Enterprise Modelling as the activity of creating a consistent set of special purpose and complementary models describing various facets of an enterprise to satisfy some purpose of some business users, [6]. Bernus and Nemes define an enterprise model as any construct on paper or in a computer or any other medium that shares some common properties with the real or contemplated system that is being modelled, [7]. Fox and Gruninger describe an Enterprise Model as a computational representation of the structure, activities, processes, information, resources, people, behaviour, goals and constraints of a business, government, or other enterprise [8]. While these definitions and descriptions highlight the different perspectives, a com-

mon theme in all of them is the use of different models such as process, information, resources, goals and other models to provide a holistic representation or description of an enterprise; in particular bringing in the business perspective together with the IT perspective. Enterprise Modelling helps focuses on the "why?" aspects of conceptual modelling [9]. This is perhaps the main distinction between IS modelling and Enterprise Modelling; a transition for the students in their understanding of models.

The state of the enterprise includes the existing processes, organizational structures and computer systems. These states are often modelled and the state of the organization is perceived (differently) by different persons through these models. This opens up for different usage areas of models as described in [2]: human sense-making, communication between people in the organization, computer-assisted analysis, quality assurance, model deployment and activation and to give the context for a traditional system development project, without being directly activated.

A perspective to Enterprise Modelling building on interactive activations is the Active Knowledge Modelling (AKM), "enabling regular industry workers to be active modellers" [10], p3. AKM aims to engage the users of models to affect the model and keep the model updated by creating models that can generate "workspaces" from which users can automatically update the models. They emphasize the importance of visual modelling and externalizing the dependencies among the different models or the aspects of an enterprise based on the knowledge of involved user, an aspect also important in participatory enterprise modelling as discussed in [11], [12].

It can be hard to judge how a model fulfills its purpose and does not deviate from its intentions. In this respect, the contributions of [12] and the Enterprise Knowledge Modelling methodology described in [13] bring relevant insight to the practice of Enterprise Modelling. Most of the effort to ensure that the models meet their purpose is the work on evaluating models. Fox and Gruninger proposed six criteria that the modelling efforts should satisfy; functional completeness, generality, efficiency, perspicuity, precision granularity and minimality [8]. Vernadat described eight principles of Enterprise Modelling, some of which are similar to the criteria proposed by Fox and Gruninger. Another approach to evaluating models is frameworks for evaluating the quality of models, e.g. the five quality criteria proposed by Bubenko et al.: ease of understanding, semantic correctness, stability, completeness and conceptual focus [9]; and SEQUAL [2]. However, these don't always ensure that the actual modelling efforts fulfill their purpose

Relative to Enterprise Modelling at the Norwegian company, Statoil, Wesenberg uses the core dimensions of SEQUAL to measure the quality of a model: syntactic quality, semantic quality and pragmatic quality. He identifies the most important aspect of a model as being pragmatic, given his goal to support sense-making and communication with models manually activated: "A model is useless unless it is understood by its target audience." [14].

3 Course Descriptions

Information Systems Modelling Advanced course is a course attended by both Masters and PhD students for one semester. These students have in their previous courses,

studied Software Engineering and Information Systems (IS) modelling through various modelling notations and languages and different aspects of IS models such as design process and requirements modelling. BPMN [15] and UML diagrams were often used. The main aim of the Masters course is to provide an overview of different aspects of modelling and to teach the students to create enterprise models. This involved the analysis of the enterprise or organizational situation, identification of the main concepts to model and modelling their dependencies. This required considering different modelling aspects studied in the current and earlier courses and putting them together as one enterprise model. This is new for the students. Hence, it was important for the teachers to take an Action Research approach to improve their understanding of their own practice as well as the students' practice of Enterprise Modelling.

Over the last two years, the course has had 15 participants per year. The course consists of lectures spread over 11 weeks, a mandatory modelling assignment, counting for 25% of the total marks for the course that the students have to deliver before the end of the semester, a presentation of the assignment to the class and a written exam, counting 75%, at the end of the semester. The modelling assignment was to create an Enterprise model of a situation that was familiar to the students. In practice, an enterprise modeller might need to be able to model a domain that they are not familiar with from the start, thus mandating an additional learning period. To be able to achieve sufficient learning during a one-semester course of enterprise modelling, it was thus important to use a familiar domain. The students were required to use the Metis Enterprise Modelling environment (a product of Troux Technologies), which provides a visual space and metamodels for creating Enterprise Models. Specific requirements were set for the assignment; they were required to:

- describe their cases in detail,
- identify the users and stakeholders of the model,
- describe the purpose of the model and to use this to define how they would evaluate their model to ensure that the model fulfilled its purpose,
- create a model that included at least five aspects of an enterprise (e.g. processes, organisation, applications),
- use the functionalities provided by Metis to selectively view the contents of the model (e.g. generate user-specific views of the model),
- evaluate their model,
- describe the lessons learned from the modelling experience.

The students were asked to present their models to the class. This is to provide the students a chance to present their work and receive feedback from the teacher as well as their peers. Since enterprise modelling is a collaborative activity conducted with the users or the domain experts, e.g. [11], [16], [12] and [13], this activity is aimed to ensure that the models that are created by the students are indeed understandable for others. At the same time, as modellers, we are expected to be able to judge when a model is sufficiently good and the added value from the model both on a short and long-term basis [17]. Hence, this exercise, although a tough step for several students, proved to be very useful for their learning.

4 An Example Model

In this section, we provide an example of an enterprise model created by one of the students as an illustration of the type of models that we expected from the students. A diverse set of models were created, e.g. the process of treating dementia patients and the process of managing the queue in a restaurant. One student modelled the communication that took place among the various departments and people in a manufacturing organisation, identifying the information that is communicated, and which communications require actions. This is shown in Figure 1. This example was motivated by the desire to understand the current situation and to suggest technologies and procedures for improving the current situation; i.e. to support informed decision making in the enterprise. The model describes the organisation structure and the roles in the organisation, the employees that fill the roles, who they communicate to, the information they communicate and the actions they may need to take. The lines across the figure are instantiations of relationships indicating how the different domains relate; e.g. which employee fills which role and the information communicated by the employee.

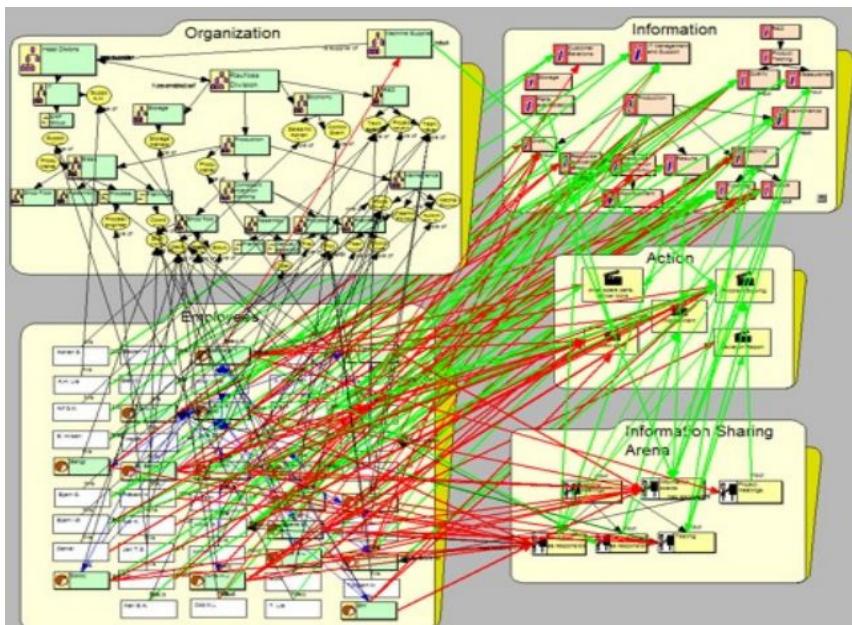


Fig. 1. Communication in a Manufacturing Enterprise

5 Observations from the Course

In this paper, we have focused on the models created by the students and how their models met the criteria presented above. The responses from the students for their modelling assignment are discussed in general as observations.

Selection of the Case to Model: About a third of the students chose a case based on their own situations such as the example presented in the previous section. The reasons that the students chose their own topics were because they had in-depth knowledge of the situation from their own experiences and they were motivated to either reflect upon past experiences or to contribute to the situation. The students also felt ownership of the knowledge which is important in learning. They believed that they could contribute to a real situation. This may be either to assess the value of modelling, which is easier in a situation familiar to them, or to convince them of the value of modelling.

Selection of Modelling Aspects: Almost all students identified five or more aspects of an enterprise in their model; the most popular ones were *organisational aspects* or an organisation chart, *processes*, *people*, *locations* (in most cases, the people in the organisations were in different geographical locations), *software applications* which are the IT systems that are required or in use, the *requirements* for these applications and *documents* which describe the applications, their requirements as well as other information. It is interesting to note that once the students started analyzing any aspect in detail, they identified additional aspects, which also often act as a link between two specific aspects. For example, in the organisation models, they identified *positions* or *roles*, which are *filled by persons*. The different aspects that were modelled were general ones in most enterprise models. However, they did explore enough to have a feel for modelling advanced concepts. For example, a challenge that was addressed by several students was the best way to model the SCRUM process in agile software development. This led to interesting discussions that no doubt contributed to the students' modelling experiences and their level of maturity as Enterprise Modellers.

Selective Views: One of the requirements of the assignment was to create selective views of model data. Students were able to identify situations in their models to extract selective contents. This is an indication that they also identified the need and value of such views as well as the types of information that users would like from their models, which is an important aspect of Enterprise Modelling. Most of the models contained a large amount of data and the different aspects and their connections can often make it difficult to highlight specific information from the model, e.g. Figure 1. Metis provides advanced functionality for creating and maintaining views. Selective views allow users to select and visualize specific information by tracing the relationships from one object or a group of objects, e.g. as shown in Figure 2, where it traces the communication of one employee with other employees, the information that is communicated and the related actions and how this information may be shared. This selective view provides an easier visual view of selective model contents that is relevant for a specific purpose.

Relationship matrices allow users to view how two sets of objects are related to one another by plotting the objects and relationships as a matrix. A relationship matrix that shows the communication among specific employees in the manufacturing organisation modelled in Figure 1 is shown in Figure 3, where the axes of the matrix show the names of the employees and the cells in the matrix show the communications. If there is an arrow, there is communication between the corresponding

employees; i.e. there is an instance of a relationship in the model between the two employees. If the cell is blank, there is no communication. The direction of the arrow indicates who is providing information to whom. A cross (x) in the matrix indicates that there is more than one instance of communication between the two employees.

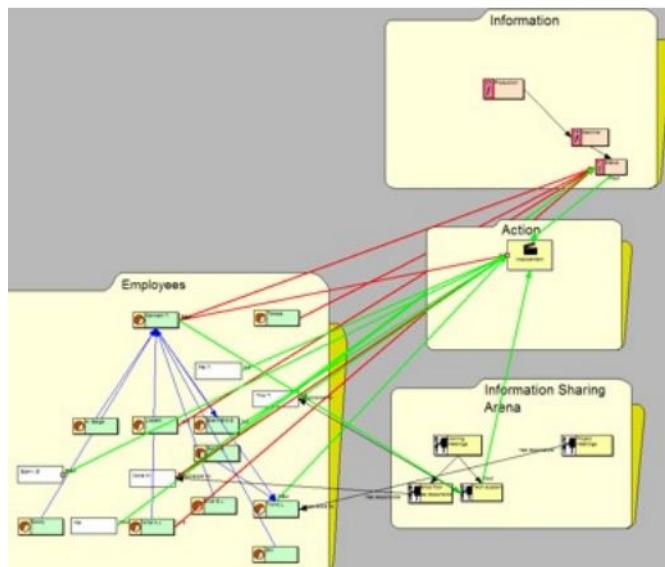


Fig. 2. Selective view: Information shared by employees

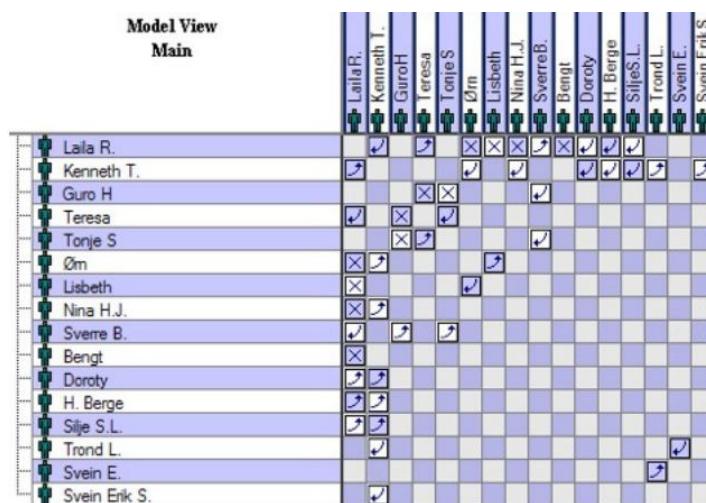


Fig. 3. Relationship matrix: communication among employees

Meta-modelling: The students were provided a few existing meta-models (modelling languages). They were required to use one of these meta-models and to enhance it if and wherever necessary and to explain the enhancements. Most of the students had some meta-model enhancements to represent specific aspects that they required and they were able to explain these. Note that in most of the modelling courses, students are taught specific modelling languages or notations such as UML or BPMN which usually does not require that the students have to think of enhancing the modelling language to include other concepts in their model.

Purpose of the Model and Evaluation: Almost all the students had a very good description of their cases and were able to describe the purposes of their models. However, they were not always able to determine how they would ensure that the purpose of the model was met and to relate the evaluation of the model to the purpose. Rather than ask the students to evaluate their model in a specific way, they were required to determine how they will evaluate their model to ensure that it meets its purpose; i.e. the evaluation method should be appropriate to the purpose of the model. Most students had evaluated their model using a formal framework such as SEQUAL or the set of characteristics by Fox & Gruninger or Vernadat or a combination of several approaches. But very few had managed to address their specific purposes and ensure that the evaluation addressed this. One reason for this is that the evaluation methods or frameworks do not address this issue adequately and lack the means to operationalize the evaluation methods.

6 Discussions and Recommendations

Moving from Information Systems Modelling to Enterprise Modelling, the students seem to have acquired an understanding of these concepts, their differences and where they can complement one another. Based on the experiences of teaching Enterprise Modelling to Information Systems students over two years, the following recommendations may be helpful for others teaching Enterprise Modelling:

- A healthy balance among the Science, Art and Practice parts of Enterprise Modelling is important. This means that in addition to the theory part or the application of the theory in the form of a modelling language or a notation, the situation analysis, which is an important part of Enterprise Modelling, should be included as a part of the course and the modelling exercises.
- To be able to conduct a good situation analysis, a holistic approach to modelling covering a number of aspects or modelling perspectives is required. For example, if the modelling exercise is focussed on developing a new IT system and developing UML models, it is equally important to understand the users' needs, why they need the IT system and how they will use it. In addition, the modelling could also support the development process and planning the development work.
- The level of maturity of the students and their experience will play a significant role in the practice of Enterprise Modelling. The students with some work experience were better at doing situation analysis and relating modelling concepts to the real world or conceptualising the real world as an Enterprise Model. Enterprise Modelling should be a course that is offered at the later stages of an Information Systems or similar study.

- It is important to anchor the modelling or relate the modelling to a domain that the students are familiar with as they may not have any experience working in a real organisation. For example, it was helpful to use software development projects and the project group as an organisation as an example.
- It is important to emphasise the users' perspectives as the modelling is conducted for the benefit of a user. Models can often become very large, although it is seldom that the whole model is important for all stakeholders. Thus it may be difficult to obtain specific information from the model relevant for the specific user. Therefore, it is important to consider how the views of the model may be extracted as well as how they can be visualised or presented to the users.

7 Summary

In this paper, we have described how Masters students at a university studying Information Systems adapt to Enterprise Modelling; expressed metaphorically, they come out of their UML tunnels and see the world outside. The paper describes an overview of the course and the observations of the course based on a modelling assignments. An example of the models created by the students is described. The main observation is that while students are able to conceptualise a situation and represent that as an Enterprise Model, they often lack the experience to clearly state the purpose of the model and therefore, lack the ability to evaluate their models appropriately. Based on the experiences of teaching Enterprise Modelling to Information Systems students, the paper also provides a set of recommendations for teachers.

The main focus so far has been on the models created by the students and to see if they are able to analyse a situation and create an enterprise model that brings together several aspects of an enterprise. We continue to improve the curriculum and teaching practice based on our observations. This year, we have focused on the enterprise modelling process followed by the students and we plan to investigate this further using a questionnaire and interviews.

Acknowledgement. The authors would like to thank Troux Technologies for providing an academic license for the Metis application that was used for the course. We thank the students and Catrine Larsson for letting us use her model as an example.

References

1. Persson, A., Stirna, J.: Why Enterprise Modelling? An Explorative Study into Current Practice. In: Dittrich, K.R., Geppert, A., Norrie, M. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 465–468. Springer, Heidelberg (2001)
2. Krogstie, J.: Model-based development and evolution of information systems: A Quality Approach. Springer, London (2012)
3. Kirkpatrick, D.L., Kirkpatrick, J.D.: Evaluating Training Programs, The Four Levels. Berrett-Koehler Publishers, Inc., San Francisco (2006)
4. Bloom, B.S.: Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain. David McKay Co Inc., New York (1956)

5. Waters-Adams, S.: Action Research in Education (2006),
[http://www.edu.plymouth.ac.uk/resined/
actionresearch/arhome.html](http://www.edu.plymouth.ac.uk/resined/actionresearch/arhome.html) (March 2013)
6. Vernadat, F.B.: Enterprise Modelling and Integration Principles and Applications. Chapman and Hall (1996)
7. Bernus, P., Nemes, L.: Organisational Design: Dynamically Creating and Sustaining Integrated Virtual Enterprises. In: IFAC World Congress, vol. A. Elsevier, London (1999)
8. Fox, M.S., Gruninger, M.: Enterprise Modeling. AI Magazine 19(3), 109–121 (1998)
9. Bubenko Jr., J.: From Information Algebra to Enterprise Modelling and Ontologies – a Historical Perspective on Modelling for Information Systems. In: Krogstie, J., Oppdahl, A., Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering. Springer (2005)
10. Lillehagen, F., Krogstie, J.: Active Knowledge Modelling of Enterprises. Springer (2008)
11. Gjersvik, R., Krogstie, J., Følstad, A.: Participatory Development of Enterprise Process Models. In: Krogstie, J., Siau, K., Halpin, T. (eds.) Information Modelling Methods and Methodologies. Idea Group Publishers (2004)
12. Persson, A., Stirna, J.: Towards Defining a Competence Profile for the Enterprise Modeling Practitioner. In: van Bommel, P., Hoppenbrouwers, S., Overbeek, S., Proper, E., Barjis, J. (eds.) PoEM 2010. LNBP, vol. 68, pp. 232–245. Springer, Heidelberg (2010)
13. Sandkuhl, K., Lillehagen, F.: The Early Phases of Enterprise Knowledge Modelling: Practices and Experiences from Scaffolding and Scoping. In: Stirna, J., Persson, A. (eds.) PoEM 2008. LNBP, vol. 15, pp. 1–14. Springer, Heidelberg (2008)
14. Wesenberg, H.: Enterprise Modelling in an Agile World. In: Johannesson, P., Krogstie, J., Opdahl, A.L. (eds.) PoEM 2011. LNBP, vol. 92, pp. 126–130. Springer, Heidelberg (2011)
15. White, S.: Business Process Modeling Notation, specification of BPMN v1.0 (2004)
16. Petersen, S.A., Bach, G., Svarlein, A.B.: Patient Care Across Health Care Institutions: An Enterprise Modelling Approach. In: van Bommel, P., Hoppenbrouwers, S., Overbeek, S., Proper, E., Barjis, J. (eds.) PoEM 2010. LNBP, vol. 68, pp. 91–105. Springer, Heidelberg (2010)
17. Krogstie, J., Dalberg, V., Jensen, S.M.: Process modeling value framework. In: Manolopoulos, Y., Filipe, J., Constantopoulos, P., Cordeiro, J. (eds.) ICEIS 2006. LNBP, vol. 3, pp. 309–320. Springer, Heidelberg (2008)

Process Mining *Versus* Intention Mining

Ghazaleh Khodabandou, Charlotte Hug, Rebecca Deneckère, and Camille Salinesi

Centre de Recherche en Informatique, Université Paris I Panthéon-Sorbonne

Ghazaleh.khodabandou@malix.univ-paris1.fr,

{Charlotte.hug, rebecca.deneckere,

camille.salinesi}@univ-paris1.fr

Abstract. Process mining aims to discover, enhance or check the conformance of activity-oriented process models from event logs. A new field of research, called intention mining, recently emerged. This field has the same objectives as process mining but specifically addresses intentional process models (processes focused on the reasoning behind the activities). This paper aims to highlight the differences between these two fields of research and illustrates the use of mining techniques on a dataset of event logs, to discover an activity process model as well as an intentional process model.

Keywords: process mining, intention mining, intentional process modeling.

1 Introduction

Tracing and analyzing activities in Information Technologies (IT) is a field that appeared when the need to discover process models emerged [1]. Process mining aims to fill the gap between activity traces obtained from event logs and process models. So far, the mined process models are activity oriented models. Dowson [2] proposed a classification of process models into activity-oriented, product-oriented and decision-oriented models. Activity-oriented process models concentrate on the activities and tasks performed in producing artifacts and their ordering. Product-oriented process models are concerned about the successive product transformations. Decision-oriented process models introduce the concept of reasoning, choice and decision-making, the processes are then seen as teleological [3,4]. [37] introduced a new category called *intentional* process models [5,52,53,54]: they take into account the notions of intention and strategies of the process enactment.

Process mining techniques focus on activities, not addressing the intentional aspect of processes. We think that intentional models are accurate to represent the users' ways of thinking and working as they capture the human reasoning behind activities. We proposed in [5,52,53,54] a new field of research dedicated to intentional process mining called *intention mining*, closely related to process mining but addressing only intentional process models.

This paper aims to highlight the differences between process mining and intention mining. We will explain, for each field of research, their objectives, the representation and models they use and some of the mining tools already in place. Furthermore, we show, with the same set of event logs, how to discover an underlying process model,

firstly an activity-oriented process model – using process mining techniques – then an intentional process model – using intention mining techniques.

This paper is organized as follows. We browse a literature overview of process mining and intention mining in Section 2 and 3. Then, we compare the two approaches by applying discovery techniques to find a process model on a predefined dataset of traces and discuss the results in section 4. Section 5 concludes this paper.

2 Process Mining

This section browses a process mining literature overview.

2.1 Objectives

The process mining idea has initially emerged in the software engineering field with Cook and Wolf [6] and applying process mining on workflow log has been proposed for the first time in [7]. It is able to “close” the Business Process Management (BPM) life-cycle that presents how a process can be managed from its definition [1], through a requirements phase, to its execution and improvement. However, in each lifecycle the requirements phase is not well supported and there is no systematic or methodical manner to diagnose the requirements. Indeed, when redesigning a process, many non-serious problems, changes or crucial information of the actual process are not taken into account to improve the process model quality. According to [1], retrieving event logs containing information about the actual process allows having an insight into the followed process model. The event logs are recorded by the Information System (IS) and are generated by the actors’ interactions with the IS. Each event in process mining is assumed to be an activity carried out by IS actors. An activity describes a well-structured step in a process. Van der Aalst proposes to classify process mining techniques into three categories [1]: Discovery, Conformance and Enhancement. In addition to these three classes, we add another emerging category: Recommendation.

- **Discovery:** Some techniques aim to discover process models by analyzing event logs. There is no a-priori information about them. For instance, event logs may be studied by α -algorithm [2] which automatically transforms them into a Petri net model presents the actors’ behaviors recorded in the event logs.
- **Conformance:** Other techniques use an a priori model to check the degree of aligning between the actual followed process model (what actors are actually performing) and the pre-defined process model. These techniques can detect the deviations (about the who, what, when and where) and the model’s intensity degree.
- **Enhancement:** It uses information recorded in event logs to improve and enrich the actual process model using methods of repair and extension. Repair has a mirror effect, (i.e. it tries to reshape the model to better illustrate reality). Extension allows widening the process model with new aspect by cross-correlating it with log.
- **Recommendation:** Some techniques aims to go a bit further about the studied process and use event logs to guess which activity may follow a current activity. [8] proposes recommendations based on URL traces. Schonenberg *et al.* propose

and experiment an approach based on recommendations which shows that the process performance is higher with an appropriate guided selection of activities [9].

2.2 Metamodels for Process Mining Results Representation

There are several metamodels for representing activity-oriented process models, such as EPCs [24], declarative models, Petri Nets BPMN, etc. However in this paper, we select only the latter two as they seem to be the most used in Process Mining.

Petri Nets. Petri nets are mathematical modeling languages allowing to model concurrency and synchronization in distributed systems. They are used as a visual communication aid to model the system behavior [10] and represent the process mining results. A Petri net is a directed graph composed of three types of components: places, transitions and arcs. Each place represents a possible system state; when occurring events or activities, transitions allow going from a place to another. Arcs maintain the relations between transitions and places.

Business Process Model and Notation (BPMN). BPMN [11] is a graphical diagram to model business processes; it aims at providing an easy graphical way to model business procedures that is understandable by all business users. Furthermore, one can model complex business process easily and map it to other languages such as BPML (Business Process Modeling Language), BPEL4WS (Business Process Execution Language for Web Services) or UML. BPMN creates a standardized link to fill the gap between business process modeling and implementation procedures. It improves the possibilities of traditional notations by managing the complex nature of internal and business-to-business processes interactions.

2.3 Process Mining Algorithms

There are various process mining algorithms that aim at discovering underlying processes from event logs. These event logs are the results of actors' interactions during the executions of tasks in different processes and contain the information about actors' behaviors, such as activities, timestamps, actors' ID, instance of process, etc. We describe briefly in the following some algorithms used in process mining.

Inference Methods. [12] compare three inference algorithms: RNet [13], Ktail [14] and Markov models [15] that infer process models with a tradeoff between accuracy and noise robustness: a) RNet is a statistical approach that characterizes current state depending on the past behaviors; b) Ktail is an algorithmic approach that evaluates the current state according to the possible future behaviors; c) Markov is a hybrid between statistical and algorithmic approaches looking at the neighboring past and future behaviors to define the future state. Later, [16, 17] proposed techniques for concurrency detection and a measure to quantify the variance between behaviors and process models.

α -Algorithm [12]. This algorithm is proposed by Van der Aalst et al. to rebuild the causality in the Petri-net workflow from the existent relations in the event log.

α -algorithm takes the event logs as input, rebuilds process models by using simple XOR, AND splits and joins; thereby creates the workflow nets as output. α -algorithm cannot handle certain constructs of workflow nets such as loops and long-term dependencies. To overcome the difficulty of α -algorithm during complex situations, an extended algorithm has been proposed: α ++ algorithm [18] that proposes new relationships between event logs to handle long-term (implicit) dependencies.

Directed Acyclic Graphs [7]. In graph theory, a directed acyclic graph is a graph that has no cycle. In process mining, the events can be transformed into dependency graphs (workflow graphs) using directed acyclic graph, representing events and their causal relations without loop. However, using this kind of graphs to model the processes is delicate as loops exist in process models. To overcome this challenge, this approach tries to count the tasks frequencies and then fold the graph. Nevertheless, the results are partially satisfying and the model does not completely match the real process.

Inductive Workflow Acquisition [19, 20]. The aim of this approach is the acquisition of workflow models and their adaptation to changing requirements by finding the best Hidden Markov Models (HMMs) that reflects the process model. One can find the HMM by merging or splitting models. Each state of HMMs corresponds to an activity node. The event logs can be observed and generated into workflow nets by inductive learning.

Hierarchical Clustering [21]. This algorithm separates a set of logs of a given process into clusters and finds the dependency graph for each log. This algorithm structures the clusters of event logs into a hierarchy tree. For each cluster, a workflow model is constructed and finally all the models are merged into a single one.

Genetic Algorithm [22]. This algorithm provides process models (Petri nets) built on causal matrix (input and output dependencies for each activity). This approach tackles problems such as noise, incomplete data, non-free-choice constructs, hidden activities, concurrency, and duplicate activities. Nevertheless, it still remains a complex task as it requires the configuration of many parameters to deal with noise and irrelevant data.

Heuristic Algorithm [23]. This approach is based on α -algorithm. It uses the likelihood by calculating the frequencies of relations between the tasks (e.g. causal dependency, loops, etc.) and construct dependency/frequency tables and dependency/frequency graphs. This approach can detect irrelevant logs. However, like the Genetic algorithm, Heuristic miner needs a complex configuration phase.

Colored Petri Nets (CPN) [24]. CPN is a graphical language for analyzing the properties of concurrent systems. CPN combines Petri nets with the CPN ML (based on functional programming language Standard ML). It can be used when concurrency and communication are crucial in the system modeling.

2.4 Process Mining Tools

Many tools emerged to support the process mining techniques. Among them, we can mention ProM [26], CPN tools [24], EMiT [27], Disco [28]. The ProM framework is a pluggable framework that supports various plugins for different techniques of process mining such as α -algorithm and its extensions. CPN tools allow modeling and analyzing CPN models [24] and simulating processes to analyze and check them. In [29,30], the authors present a combination of CPN tools and ProM framework plugin implemented to improve business processes modeling. Disco allows to automatically map the event logs with CSV and XLS extensions to the appropriate XES or MXML notations which are supported by ProM and to have an insight into the process from event logs very quickly. It optimizes performance, controls deviations and explores variations. However, it is a commercial tool and does not provide information about the used algorithms. EMiT is able to integrate timing information using an extended version of α -algorithm. This tool transforms the event log of commercial systems to XML format and mines to find the causal relations between logs and based on that, rebuilds a Petri net represented in a graphical model.

3 Intention Mining

This section gives a literature overview of intention mining.

3.1 Objectives

The definition of “intention” according to [31] is: “a determination to act in a certain way; a concept considered as the product of attention directed to an object or knowledge [...].” From a psychological point of view, intention is defined as follows: “Our common sense psychological scheme admits of intentions as states of mind; and it also allows us to characterize actions as done intentionally, or with a certain intention” [32]. Purpose in an IS context is also essential for any organization. IS are created to fulfill organization needs and their functionalities and properties are defined according to the objectives of the organization. According to [33], an intention is “an optative” statement, a state or a result that is expected to be reached or maintained in the future.

Some approaches called Intention Mining have been defined, however their analysis is not based on traces of IS activities; [34]’s approach uses classification techniques to classify home video content. The analysis is based on what is recorded with camcorders and provides categories to sort videos. They do not provide process models but some of their classification techniques could be reused in process models context.

In our point of view, Intention Mining aims at extracting sequences of actors’ activities from sets of event logs to infer related actors’ intentions. A set of activities corresponds to the achievement of an intention. Intention mining uses event logs as input and produces intentional process models. It is a field related to intentional process models. As for process mining techniques, Intention Mining tackles the four challenges of discovery, conformance, enhancement and recommendation:

- Discovery: Identifying the underlying actors' intentions and strategies from the event logs allows defining intentional process models.
- Conformance: Checking the conformity between a prescribed intentional model and its enactment allows measuring the gap between the prescriptions and what is actually done by users.
- Enhancement: The conformance checking allows identifying the distance between the model and the traces, which helps to define what is wrong with the model (which intention is never achieved, which strategy is never used, etc.).
- Recommendation: Using the event logs repository and the discovered intentional process models allows providing recommendations to IS actors at run-time, based on their supposed reasoning behind their activities.

3.2 Metamodels for Intention Mining Results Representation

Processes may be formalized in an intentional way. The common aim of goal-modeling approaches is to model the processes according to the purpose of the actors/projects/organizations. We quote among them i* [35], KAOS [36] and Map [37].

KAOS. This approach proposes to specify the system and its environment by a requirements model instance of a metamodel to support the goals, agents, and alternatives. It is based on a goals diagram where goals are related together through AND/OR decomposition links. KAOS uses goals to specify, analyze, negotiate, document and modify the systems requirements. To do so, the decompositions refine high-level goals identified by actors into thinner particle of goals. This refinement requires classifying goals according to their level of abstractions and linking the same goals at the same level of abstraction. This approach supports variability and have a well-structured semantic but is less involved in the intentional aspect of IS actors. Furthermore, KAOS has a rigid task-decomposition; modeling complex intentional processes is then difficult [28].

I*. The i* framework is a modeling language that aims at analyzing IS and the environments of organizations to model processes by focusing on the relationships between actors and theirs goals. It consists in two main models: the strategic dependency model (SD) and the strategic rational model (SR). The SD describes external relationships between actors (called strategic actors). The SR describes the internal relationships between actors. The i* framework is used to model the business strategy with organizational strategic goals. i* supports actor-oriented and goal-oriented concepts. The actor-oriented concept allows modeling requirements of IS by concentrating on the dependencies between actors' goals. Actors are autonomous entities with uncontrollable and non-cognizable behaviors. They are different and independent in their ways of reasoning and consequently have diverse goals. i* models aim at producing a conceptual framework to represent the processes involving agents, i.e., software actors and software systems. i* is able to assess the functional or non-functional requirements of systems so it can capture what, how and why a software component is developed. It recommends the use of the notion of non-functional requirements using "soft goals" identified as evaluation criteria. The alternatives then contribute to different degrees of satisfaction of these goals. How-

ever, this modeling language has an operational semantic for the tasks but not for the goals and it is not used to model strategic goals. i* is not designed to be a variable framework therefore, it does not afford a high level of flexibility.

Map. This modeling language is an intentional process metamodel that allows formalizing flexible processes. Map supports variability for the goals and offers the possibility to follow different strategies by focusing on the intentional aspect when enacting methodological processes. During its enactment, a process is not limited to linear activities; actors, according to their context, have a variety of choices to execute a task. Map models (instances of Map metamodel) guide the actors by proposing dynamic choices according to their intentions. Map models can be executed non-sequentially and followed until the fulfillment of intentions. Thereby, map process models offer a better adaptability to the context of each actor. Map specifies processes in a flexible way by focusing on the intentions and the different ways to fulfill them. A map model is presented as a graph which nodes represent intentions and edges represent strategies. An edge is defined between two nodes where its related strategy can be used to achieve the intended target node. There may be several edges entering a node representing all the strategies to fulfill an intention. Fulfilling a specific intention with a particular strategy is related to a specific guideline defining the activities to perform. This model allows describing high level organizational intentions. The intentional Map metamodel has been introduced in the IS engineering domain [37] and was validated in several works: requirement engineering [38], method engineering [39], and enterprise knowledge development [40].

3.3 Intention Mining Algorithms

In order to retrieve the intentions from the traces of actors' activities and due to the variability of traces in terms of length and nature, we propose some algorithms based on probabilistic and statistical techniques. The probabilistic models provide the information about the nature of data (i.e. if the observed data is a harvest of hazard or an intended result). Moreover, they model the data taking into account their temporal aspect. Since the observed side of data could hide the latent one, the probabilistic models can also formalize this concealed side and extract the characteristics of both observed and latent data. Hidden Markov Models (HMMs) [41] seem promising to mine intentions [52,54](concurrency can be handled in future extensions). Hereafter, we describe some algorithms used in HMMs which help mining intentions.

Viterbi Algorithm (VA) [42]. VA is commonly used to decode convolutional codes (to correct errors in noisy channels) and in the context of HMMs. It finds the most likely sequence of hidden states (Viterbi path) for a given observed sequence using trellis, which is a type of Finite States Machine (FSM) with states and transitions. In HMMs, the observations are generated by the underlying states; HMMs enable to find the hidden structure by estimating the parameters of observations sequences. Given an observed sequence, the VA uses various metrics to evaluate which path is the most likely one. In the context of Intention mining, the hidden states are the IS actors' intentions which generate the observations (actors activities' sequences). To find the correct path, we can use a brute-force method but the complexity of computation C^n

can explode with the increase of the length of observations n for the number of states of C . Nevertheless, the VA allows to compute the Viterbi path with a nC^2 complexity, which is considerably lower than C^n . On the other hand, VA minimizes the error probability of states transitions by determining the most likely set of states. To infer the most likely set of intentions, the VA requires knowing the parameters of observations, i.e. the transition probabilities, initial probabilities of states and emission probabilities. These parameters can be estimated by Maximum-Likelihood Estimation (MLE). The transition probabilities can be calculated by counting the average frequency of states transition from one intention to another. We calculate the emission probabilities by counting the average apparition frequencies for a given activities sequence for a given intention. Thus, the VA is applicable when this information is available; this is possible in the case of supervised learning. Moreover, the VA is particularly useful when activities can belong to several intentions.

Baum-Welch Algorithm (BWA) [43]. BWA is a special kind of Generalized Expectation-Maximization (GEM) algorithm [44]. In the case of unsupervised learning, the BWA allows computing the unknown parameters of HMMs. To do this, the BWA uses Forward-Backward algorithm [45]. The use of iterative algorithms such as the EM algorithm allows estimating the optimal solution fairly, accurately and quickly. As mentioned in the VA, if the parameters of observations sequences are known, the computation of these parameters is a simple task. Nevertheless, since in some cases we do not know this a priori information for the observed sequences, we cannot directly count the frequencies from data, therefore, the BWA uses Forward-Backward algorithm to estimate the expected frequencies.

3.4 Intention Mining Tools

There is no intention mining tool to our knowledge, as intention mining has not been applied yet to re-build intentional process models and discover goals behind activities. We aim to provide a tool that will implement the developed algorithms.

4 Case Study

In this section we illustrate our comparison with the use of process mining and intention mining techniques on the same dataset.

4.1 Dataset

The dataset, obtained from [46], contains the event log of claims handling in an insurance company. These claims are supported by two call centers in two different locations: Sydney and Brisbane. The log contains 46 138 events and 3 512 cases (claims). The incoming calls volume and average total call handling time are the same in the two centers. However, call center agents handle differently the incoming calls. The agents first handle the claims; the rest of the process is treated in the back-office of the insurance company. According to [46], although this dataset is synthetic with non-noisy event log, the α -algorithm cannot mine correctly and extract the right

model. In Equation (1), A represents the transition matrix of the activities: they are ordered from left to right and top to down: Incoming claim (A1) (first line, fist column), Brisbane checks if sufficient information is available (A2), Brisbane registers claim (A3), Sydney checks if sufficient information is available (A4), Sydney registers claim (A5), Determine likelihood of claim (A6), Assess claim (A7), Advise claimant on reimbursement (A8), Initiate payment (A9), Close claim (A10) and End (A11). This matrix shows the following activities for each activity, and in which proportion; for instance, A1 activity is followed at 48,97% by A2 activity and by A4 activity at 51,03% .

$$A = \begin{pmatrix} 0 & 0.4897 & 0 & 0.5103 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8866 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1134 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7952 & 0 & 0 & 0 & 0 & 0.2048 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.8342 & 0 & 0 & 0.1658 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4039 & 0.3990 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5030 & 0.2389 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2389 & 0 & 0.2581 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2581 & 0 & 0.7419 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (1)$$

4.2 Process Mining Result

We choose to illustrate the process mining results with a Petri net representation of the process model found with α -algorithm (Figure 1). It was obtained using ProM tool by choosing the causal dependency parameter. The choice of this parameter is due to the fact that Petri nets allow the representation of causal dependencies between activities occurrences to model reactive systems [47].

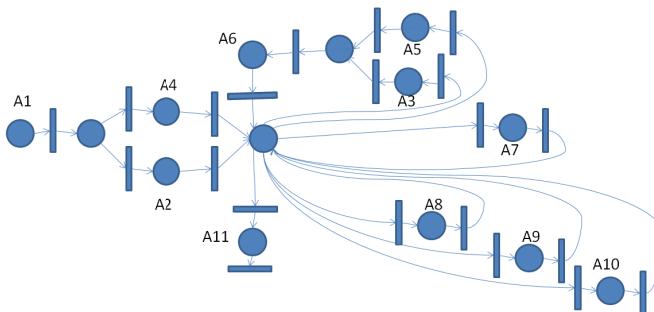


Fig. 1. The process model represented as a Petri net

When comparing the links between the activities in (1) and the Petri net obtained from ProM, we perceive that there are transitions allowed in Petri net but not in matrix A (1). For instance, Petri net allows the sequence A5, A6, A7, A5, A6 whereas this sequence is not possible in A since there is no transition between A7 and A5. This means that both Petri net and the transition matrix fit the dataset, but Petri net is more general and allows cases that are not present in the dataset, contrary to the transition matrix. Thus, Petri net, for this dataset, suffers from an under-fitting problem.

4.3 Intention Mining Result

In this paper, we choose the Map representation to illustrate the intentional process model found by intention mining. i* and KAOS will be studied further in our next works on intention mining. They focus on the operational level rather than the organization level and do not raise the issue of alignment [48], whereas Map allows working on strategic goals. As mentioned earlier in section 3.3, both VA and BWA are able to associate a sequence of intentions to a sequence of activities. However, these two algorithms need to be fed with the set of intentions, which is not the case here as this set is not yet defined. We then implement a new algorithm to find (a) the groups of strategies used by the agents of the insurance company during the claims treatments, and (b) the groups of intentions linked to those strategies.

Strategies Miner Algorithm. To construct the map process model, we have to find the strategies that the agents of the insurance company use to fulfill their intentions. To do this, we define a strategy by the triplet $\{s, \mathcal{A}, e\}$, where s is the starting activity, e is the ending activity and \mathcal{A} is a set of activities in between. The realization of this strategy is a sequence of activities starting with s , followed by a sequence of activities comprising all the elements in \mathcal{A} and finishing with e . We have developed two simple rules to organize the activities into strategies. The first rule is based on what we call a bottleneck activity. A bottleneck activity a is an activity for which there are at least two possible preceding activities, and for every possible preceding activity the transition to activity a occurs with a probability of 1. Of course, the initial and final activities are excluded from this definition. Then, a bottleneck activity can only be at the beginning of a strategy. The second rule is that for any strategy, there is a sequence of activities such that the probability of having this sequence is higher than a threshold β . The first rule accounts for the fact that if there are compulsory activities, these activities do not represent different strategies. The second rule accounts for the fact that the activities composing a strategy have to be frequently used sequentially. Note that for some strategies, the order of activities is not significant since the activities can be performed simultaneously or in a random order. However, the first and the last activities of the sequence must always be the same. Thereby, we can group the activities into strategies. We say that these rules define a set of strategies with parameter β . The strategies miner algorithm is given as follows in pseudo-code.

```

Inputs: cases
Outputs: strategies
Begin
    Make a strategy out of each case
    Merge equal strategies
    Divide strategies at the bottleneck(s)
    Merge equal strategies
    For each strategy {s,  $\mathcal{A}$ , e}
        if max  $\mathbb{P}(\mathcal{A}, e | s) < \beta$  then
            divide strategy
        endif
    EndFor
End

```

Table 1 presents the strategies obtained with threshold $\beta = 0.01$ and $\beta = 0.3$. The column *probability* indicates the probability of having the corresponding strategy among the cases of the dataset.

Table 1. Discovered strategies with threshold $\beta = 0.01$ and $\beta = 0.3$

Strategy index	$\beta=0.01$				$\beta=0.3$			
	Starting activity	Ending activity	Set in between	Probability	Starting activity	Ending activity	Set in between	Probability
1	A1	A11	A3	0.0555	A1	A3	A2	0.4342
2	A1	A11	A4	0.1045	A1	A5	A4	0.4058
3	A1	A2	A3	0.4342	A1	A4	none	0.5103
4	A1	A5	A4	0.4058	A11	None	none	1.0000
5	A6	A11	none	0.1392	A1	A2	none	0.4897
6	A6	A11	A7	0.1381	A6	none	none	0.8400
7	A6	A11	A7,A8,A9,A10	0.5626	A6	A7	none	0.7007
8	none	none	none	none	A8	A11	A9, A10	0.5626

The algorithm discovers 7 strategies with $\beta = 0.01$ and 8 strategies using $\beta = 0.3$. With $\beta = 0.3$, we find that the first strategy begins with A1 activity (incoming claim), followed by A2 activity (Brisbane checks if sufficient information is available) and ends with A3 activity (Brisbane registers claim) with a probability of 0.4342.

We evaluated the discovered strategies with different values of β (from 0.01 to 0.9). The adjustment of the threshold to 0.3 is justified by the two following constraints: (1) the lower the value of β , the higher the number of activities per strategy, which is more interesting at the intentional level. (2) β is chosen such that no strategy contains more activities than half of the total activities number (here 5,5). For instance, in table 1 when $\beta = 0.01$, strategy 7 represents a set of six activities which is more than half of initial activities. Therefore, $\beta = 0.3$ satisfies these two constraints as the strategies found by this threshold are all inferior to 5.5 activities.

The next step is to infer the intentions from these strategies. However, even if this algorithm is able to find the groups of strategies, for now it is not able to find the intentions. In this paper, the intentions and the links between them (step b) will be determined by human inference and reasoning. In the future, we will automate this step using ontologies of activities and natural language analysis techniques [51]. The results are given in next section.

Process Modeling Using Map. Figure 2 presents the intentional Map process model manually built from the strategies found in Table 1 with $\beta = 0.3$. By inference, we determine intentions that can be fulfilled by strategies that have the same nature and we name the strategies according to the activities.

Strategies 1, 2, 3 and 5, for instance, will be four ways to reach the same intention, which is *Validate the claim*. Strategies 1 and 2 show that there are two ways to validate the claim: either by *Brisbane checking* or *Sydney checking*. Strategies 3 and 5 correspond to the same strategies 1 and 2, as they correspond to a control activity that fails to validate the claim (the intention *Validate the claim* is not achieved). Strategy 7

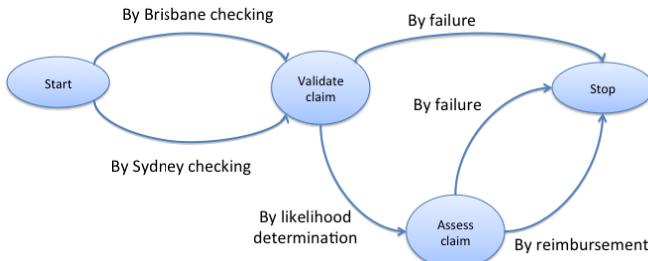


Fig. 2. The process model represented as intentional map

corresponds to the set of activities that will allow to *Assess the claim* by *likelihood determination*. As for strategies 3 and 4, strategy 6 corresponds to a set of activities that does not allow reaching an intention, as there is an activity that fails (the claim cannot be assessed as the likelihood determination failed). Strategy 8 allows reaching the *stop the process* intention after the *reimbursement* of the client (note that activities A9 and A10 can be enacted in any order). Strategy 4 corresponds to the activities that end the process, so the intention reached in this case is *stop the process* when there has been a *failure* in the normal claim process (either in the claim checking or in the likelihood determination). This step will be automated in future works.

4.4 Discussion

This case study shows that either process mining or intention mining allows discovering a process model from a set of event logs. Process mining offers really good techniques to mine activity process models, not only to discover the models, but also to define the gap between the models and traces, and even to make recommendation about the possible following activities at run-time. However, these techniques focus only on the operational aspect of the process (activity oriented).

We strongly believe that intentional process models are an adequate formalism to guide users through the enactment of their activities. They allow modeling processes in a flexible way (the notion of sequence does not exist), variability can be introduced (alternatives path can be followed, different strategies can be used to achieve the same intention), thus, we think this kind of model allows more creativity than activity oriented process models. Intention mining for intentional process modeling is a new field of research and techniques are only emerging, however, we think that it will offer promising concepts and tools to mine intentional process models. Using intention mining techniques will help to promote this type of process models, as discovering them will be easier (as shown in the case study). Moreover, the techniques and algorithms that we are currently defining will also help to recommend tasks, based on the supposed reasoning of the users (theirs intentions), as close as possible to human ways of thinking and working.

We think that intention mining will not be hampered by the same problems identified in process mining [50], as intentional process models are flexible. For instance,

the problems of hidden tasks (no-recorded tasks) or duplicate tasks (a process model with same task twice) should be overcome with an intentional modeling, where activities are of less importance with a representation on a higher level. The concept of loop is also usual in intentional modeling - for instance in map process models, a section can be enacted several times, until the intention is achieved - whereas it is often a difficult problem to handle in process mining. However, intention mining is not the answer to these process mining problems, it is only another field of mining research, aiming to work on another kind of process models. It will have problems on its own: for instance, to define concurrent and exclusive strategies to achieve the same intention will be quite a problem to solve automatically, without human-expert inference.

5 Conclusion

In this paper, we presented the objectives, algorithms, models and tools for process mining and intention mining. The developed case study showed that, on this specific dataset, either process mining or intention mining techniques allow discovering a process model. However, intention mining (for intentional process models) is a recent emerging research field and a lot of work is still needed to develop full algorithms.

Our next step will be to automate the merging of strategies to define intentions and name them using ontologies and natural language analysis techniques. We will also provide recommendations using map process models and traces.

References

1. Van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st edn., Berlin (2011)
2. Dowson, M.: Iteration in the Software Process. In: Proc. 9th Int. Conf. on Soft. Eng. (1998)
3. Veblen, T.: Why is Economics not an Evolutionary Science? *The Quarterly Journal of Economics* 12(4), 373–397 (1898)
4. Ralph, P., Wand, Y.: A Teleological Process Theory of Software Development. *JAIS Theory Development Workshop* 8(23) (2008)
5. Hug, C., Deneckère, R., Salinesi, C.: Map-TBS: Map process enactment traces and analysis. In: Procs. of RCIS 2012, Valencia, Spain, pp. 204–209 (2012)
6. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. *ACM Trans. on Soft. Engineering and Methodology* 7(3), 215–249 (1998)
7. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process Models from Workflow Logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
8. Mobasher, B., Cooley, R., Srivastava, J.: Automatic Personalization Based On Web Usage Mining. *Communication of ACM* 43(8), 142–151 (2000)
9. Schonenberg, H., Weber, B., van Dongen, B.F., van der Aalst, W.M.P.: Supporting Flexible Processes through Recommendations Based on History. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
10. Van der Aalst, W.M.P., Stahl, C.: Modeling Business Processes: A Petri Net Oriented Approach. MIT Press, Cambridge (2011)

11. <http://www.bpmn.org> (consulted February 1, 2013)
12. Cook, J., Wolf, A.: Automating process discovery through event-data analysis. In: Proceed. of the 17th Int. Conf. on Software Engineering, pp. 73–82. ACM Press (1995)
13. Das, S., Mozer, M.C.: A unified Gradient Descent/Clustering Architecture for Finite State Machine Induction. In: Proc. of the 1993 Conf., Advances in Neural Information Processing system, vol. 6, pp. 19–26. Morgan Kaufmann (1994)
14. Biermann, A.W., Feldman, J.A.: On the Synthesis of Finite States Machines from Samples of Their Behavior. *IEEE Transactions on Computers* 21(6), 592–597 (1972)
15. Baum Leonard, L.E., Petrie, T.: Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Math. Stat.* 37(6), 1554–1563 (1966)
16. Cook, J.E., Wolf, A.L.: Event-based detection of concurrency. In: Proceed. of the 6th Int. Symposium on the Foundations of Software Engineering (FSE-6), pp. 35–45 (1998)
17. Cook, J.E., Wolf, A.L.: Software process validation: quantitatively measuring the correspondence of a process to a model. *ACM Transactions on Software Engineering and Methodology* 8(2), 147–176 (1999)
18. Wen, L., Wang, J., Sun, J.G.: Detecting Implicit Dependencies between Tasks from Event Logs. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 591–603. Springer, Heidelberg (2006)
19. Herbst, J., Karagiannis, D.: Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. In: Proceed. of the 9th Int. Workshop on Database and Expert Systems Applications, pp. 745–752 (1998)
20. Herbst, J.: Dealing with concurrency in workflow induction. In: European Concurrent Engineering Conference, SCS Europe (2000)
21. Greco, G., Guzzo, A., Pontieri, L.: Mining Hierarchies of Models: From Abstract Views to Concrete Specifications. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 32–47. Springer, Heidelberg (2005)
22. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.: Genetic Process Mining. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 48–69. Springer, Heidelberg (2005)
23. Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integ. Computer-Aided Engineering* 10(2), 151–162 (2003)
24. Van der Aalst, W.M.P.: Formalization and verification of event-driven process chains. *Information and Software Technology* 41(10), 639–650 (1999)
25. Jensen, K., Kristensen, L.M.: Coloured Petri Nets. Springer, Berlin (2009)
26. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The proM framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
27. van der Aalst, W.M.P., van Dongen, B.F.: Discovering workflow performance models from timed logs. In: Han, Y., Tai, S., Wikarski, D. (eds.) EDCIS 2002. LNCS, vol. 2480, pp. 45–63. Springer, Heidelberg (2002)
28. Thevenet, L.-H., Salinesi, C.: Aligning IS to organization's strategy: The INSTAL method. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 203–217. Springer, Heidelberg (2007)
29. Medeiros, A., Günther, C.W.: Process mining: Using CPN tools to create test logs for mining algorithms. In: Procs. of the 6th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, pp. 177–190 (2005)
30. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering colored Petri nets from event logs. *International Journal STTT* 10(1), 57–74 (2008)
31. Merriam-Webster Dictionary. Merriam-Webster (2002)

32. Bratman, M.E.: Intention, Plans, and Practical Reason. Harvard University Press (1987)
33. Jackson, M.: Software Requirements & Specifications, a Lexicon of Practice, Principles and Prejudices. ACM Press, Addison-Wesley (1995)
34. Mei, T., Hua, X.S., Zhou, H.Q., Li, S.: Modeling and Mining of Users' Capture Intention for Home Videos. *IEEE Transactions on Multimedia* 9(1), 66–77 (2007)
35. Yu, E.: Modelling Strategic Relationships for Process Reengineering, PhD thesis, University of Toronto, Dpt. of Computer Science (1995)
36. Dardenne, A., van Lamsweerde, A., Fickasu, S.: Goal-directed requirements acquisition. *Sci. Comput. Program* 20(1-2) (1993)
37. Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modeling. *Requirements Engineering* 4(4) (1999)
38. Prakash, N., Rolland, C.: Systems Design for requirements expressed as a map. In: Proceed. of the Conference IRMA 2006, Washington, DC (2006)
39. Kornyshova, E., Deneckère, R., Salinesi, C.: Method Chunks Selection by Multicriteria Techniques: an Extension of the Assembly-based Approach. In: ME 2007, Switzerland (2007)
40. Barrios, J., Nurcan, S.: Model Driven Architectures for Enterprise Information Systems. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 3–19. Springer, Heidelberg (2004)
41. Juang, B.H., Rabiner, L.R.: A Probabilistic Distance Measure for Hidden Markov Models. *AT&T Technical Journal* 64(2), 391–408 (1985)
42. Forney, G.D.: The Viterbi Algorithm. *Proceeding IEEE* 61(3), 268–278 (1973)
43. Baum, L.E., Petrie, T., Soules, G., Weiss, N.: A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *Ann. Math. Statist.* 41(1), 164–171 (1970)
44. Moon, T.K.: The expectation-maximization algorithm. *IEEE Signal Processing Magazine* 13(6), 47–60 (1996)
45. Yu, S.Z., Kobayashi, H.: An efficient forward-backward algorithm for an explicit-duration hidden Markov model. *IEEE Signal Processing Letters* 10(1), 11–14 (2003)
46. <http://www.processmining.org> (consulted February 1, 2013)
47. van Glabbeek, R.J., Goltz, U., Schicke, J.-W.: On Causal Semantics of Petri Nets. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 43–59. Springer, Heidelberg (2011)
48. Bleistein, S.J., Cox, K., Verner, J., Phalp, K.: B-SCP: A requirements analysis framework for validating strategic alignment of organizational IT based on strategy, context, and process. *Information and Software Technology* 48(9), 846–868 (2006)
49. Van der Aalst, W.M.P., Weijters, A.J.M.M.: Process mining: a research agenda. *Computers and Industry* 53(3), 231–244 (2004)
50. Van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering* 16(9), 1128–1142 (2004)
51. Lallé, S., Luengo, V., Guin, N.: Méthode semi-automatique pour le développement d'un ensemble de techniques de diagnostic des connaissances. In: Conférence EIAH (2013)
52. Khodabandou, G., Hug, C., Deneckère, R., Salinesi, C.: Supervised Intentional Process Models Discovery using Hidden Markov Models. In: Procs. RCIS 2013, Paris, France (2013)
53. Khodabandou, G., Hug, C., Deneckère, R., Salinesi, C., Bajec, M., Kornyshova, E., Janković, M.: COTS Products to Trace Method Enactment: Review and Selection. In: Procs. of ECIS 2013, Utrecht, Netherlands (2013)
54. Khodabandou, G.: Contextual Recommendations Using Intention Mining on Process Traces. In: Procs. of RCIS 2013 Doctoral Consortium, Paris, France (2013)

Author Index

- Adam, Sebastian 78
Alexopoulou, Nancy 18
Anwar, Adil 426
Ayora, Clara 246
Aysolmaz, Banu 154

Baier, Thomas 109
Baresi, Luciano 214
Bider, Ilia 63
Boubaker, Anis 139
Brinkkemper, Sjaak 370

Castro, Jaelson 184
Charif, Yasmine 139

Dar, Zakria Riaz 63
de Kinderen, Sybren 339
de la Vara, Jose Luis 168
Demirörs, Onur 154
Deneckère, Rebecca 466

España, Sergio 184

Fernández-Ropero, María 94
Ferreira, Diogo R. 124

Gopalakrishnan, Sundar 324

Hadar, Irit 2
Haisjackl, Cornelia 2
Halpin, Terry 308
Henderson-Sellers, Brian 1
Hoppenbrouwers, Stijn 385
Hug, Charlotte 466

Idani, Akram 426
Inglés-Romero, Juan F. 441
İren, Deniz 154

Jalali, Amin 199
Jansen, Slinger 370
Johannesson, Paul 199

Khodabandelou, Ghazaleh 466
Kirsch-Pinheiro, Manuele 32

Koch, Matthias 78
Krogstie, John 324, 395, 456

Lagarde, Remco 385
Ledru, Yves 426
Le Grand, Bénédicte 32
Leopold, Henrik 292
Leshob, Abderrahmane 139
Lespérance, Yves 277
Lohrmann, Matthias 230
Lotz, Alex 441

Marconi, Annapaola 214
Marrella, Andrea 277
Mendling, Jan 109, 292
Mili, Hafedh 139

Nikolaidou, Mara 18

Oldenhove, Danny 385
Oliveira, Karolyne 184
Outmazgin, Nesi 48

Pastor, Oscar 168, 184
Pelechano, Vicente 246
Pérez-Castillo, Ricardo 94
Perjons, Erik 63
Petersen, Sobah Abbas 456
Piattini, Mario 94
Pinggera, Jakob 2, 261
Pistore, Marco 214
Pittke, Fabian 292
Plataniotis, Georgios 339
Proper, Henderik A. 339

Ralha, Célia G. 124
Reichert, Manfred 2, 230, 246, 261
Reijers, Hajo A. 94
Reinhartz-Berger, Iris 354
Riegel, Norman 78
Rychkova, Irina 32

Salinesi, Camille 466
Sánchez, Juan 168
Schlegel, Christian 441

- Sedrakyan, Gayane 411
Sindre, Guttorm 324
Sirbu, Adina 214
Snoeck, Monique 411
Soffer, Pnina 2, 48
Stary, Christian 18
Szimanski, Fernando 124

Torres, Victoria 246, 261

van der Weide, Theo 385
van Tuijl, Geurt 370
Vicente-Chicote, Cristina 441
Vlaanderen, Kevin 370

Wagner, Gerd 124
Weber, Barbara 2, 246, 261
Wulf-Hadash, Ora 354

Zugal, Stefan 2