

LNCS 8841

Robert Meersman Hervé Panetto
Tharam Dillon Michele Missikoff
Lin Liu Oscar Pastor Alfredo Cuzzocrea
Timos Sellis (Eds.)

On the Move to Meaningful Internet Systems: OTM 2014 Conferences

Confederated International Conferences:
CoopIS and ODBASE 2014
Amantea, Italy, October 27–31, 2014, Proceedings



Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Robert Meersman Hervé Panetto
Tharam Dillon Michele Missikoff
Lin Liu Oscar Pastor Alfredo Cuzzocrea
Timos Sellis (Eds.)

On the Move to Meaningful Internet Systems: OTM 2014 Conferences

Confederated International Conferences:
CoopIS and ODBASE 2014
Amantea, Italy, October 27-31, 2014
Proceedings



Springer

Volume Editors

Robert Meersman, TU Graz, Austria
E-mail: rameersman@gmail.com

Hervé Panetto, University of Lorraine, Vandoeuvre-les-Nancy, France
E-mail: herve.panetto@univ-lorraine.fr

Tharam Dillon, La Trobe University, Melbourne, VIC, Australia
E-mail: tharam.dillon7@gmail.com

Michele Missikoff, IASI - CNR, Rome, Italy
E-mail: michele.missikoff@iasi.cnr.it

Lin Liu, Tsinghua University, Beijing, China
E-mail: linliu@tsinghua.edu.cn

Oscar Pastor, Universidad Politécnica de Valencia, Spain
E-mail: opastor@dsic.upv.es

Alfredo Cuzzocrea, University of Calabria, Rende, Italy
E-mail: cuzzocrea@si.dimes.unical.it

Timos Sellis, RMIT University, Melbourne, VIC, Australia
E-mail: timos.sellis@rmit.edu.au

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-662-45562-3 e-ISBN 978-3-662-45563-0
DOI 10.1007/978-3-662-45563-0
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014953775

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

© Springer-Verlag Berlin Heidelberg 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, 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.

protective laws and regulations and therefore free for general use. While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typeetting: 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)

General Co-Chairs' Message for OnTheMove 2014

The OnTheMove 2014 event held during October 27–31 in Amantea, Italy, further consolidated the importance of the series of annual conferences that was started in 2002 in Irvine, California. It then moved to Catania, Sicily in 2003, to Cyprus in 2004 and 2005, Montpellier in 2006, Vilamoura in 2007 and 2009, in 2008 to Monterrey, Mexico, to Heraklion, Crete in 2010 and 2011, to Rome 2012, and to Graz in 2013. This prime event continues to attract a diverse and relevant selection of today's research worldwide on the scientific concepts underlying new computing paradigms, which of necessity must be distributed, heterogeneous, and supporting an environment of resources that are autonomous yet must meaningfully cooperate. Indeed, as such large, complex, and networked intelligent information systems become the focus and norm for computing, there continues to be an acute and even increasing need to address the implied software, system, and enterprise issues and discuss them face to face in an integrated forum that covers methodological, semantic, theoretical, and application issues as well. As we all realize, email, the Internet, and even video conferences on their own are not optimal nor even sufficient for effective and efficient scientific exchange.

The OnTheMove (OTM) Federated Conference series has been created precisely to cover the scientific exchange needs of the communities that work in the broad yet closely connected fundamental technological spectrum of Web-based distributed computing. The OTM program every year covers data and Web semantics, distributed objects, Web services, databases, information systems, enterprise workflow and collaboration, ubiquity, interoperability, mobility, grid, and high-performance computing.

OnTheMove does *not* consider itself a so-called multi-conference event but instead is proud to give meaning to the “federated” aspect in its full title¹: It aspires to be a primary scientific meeting place where all aspects of research and development of Internet- and intranet-based systems in organizations and for e-business are discussed in a scientifically motivated way, in a forum of loosely interconnected workshops and conferences. This year's 11th edition of the OTM Federated Conferences event therefore once more provided an opportunity for researchers and practitioners to understand, discuss, and publish these developments within the broader context of distributed, ubiquitous computing. To further promote synergy and coherence, the main conferences of OTM 2014 were conceived against a background of three interlocking global themes:

¹ On The Move Towards Meaningful Internet Systems and Ubiquitous Computing—Federated Conferences and Workshops

- Trusted Cloud Computing Infrastructures Emphasizing Security and Privacy
- Technology and Methodology for Data and Knowledge Resources on the (Semantic) Web
- Deployment of Collaborative and Social Computing for and in an Enterprise Context

Originally the federative structure of OTM was formed by the co-location of three related, complementary, and successful main conference series: DOA (Distributed Objects and Applications, held since 1999), covering the relevant infrastructure-enabling technologies, ODBASE (Ontologies, DataBases and Applications of SEmantics, since 2002) covering Web semantics, XML databases and ontologies, and of course CoopIS (Cooperative Information Systems, held since 1993), which studies the application of these technologies in an enterprise context through, e.g., workflow systems and knowledge management. In the 2011 edition of DOA security aspects, originally started as topics of the IS workshop in OTM 2006, became its focus as secure virtual infrastructures. Subsequently these further broadened to include Cloud-based systems emphasizing aspects of trust and privacy. As these latter aspects came to dominate agendas in its own and overlapping research communities, we decided for 2014 to rename the event as the Cloud and Trusted Computing (C&TC) conference, and to organize and launch it in a workshop format to define future editions.

Both main conferences specifically seek high-quality contributions of a more mature nature and encourage researchers to treat their respective topics within a framework that simultaneously incorporates (a) theory, (b) conceptual design and development, (c) methodology and pragmatics, and (d) application in particular case studies and industrial solutions.

As in previous years we again solicited and selected additional quality workshop proposals to complement the more mature and “archival” nature of the main conferences. Our workshops are intended to serve as “incubators” for emergent research results in selected areas related, or becoming related, to the general domain of Web-based distributed computing. This year this difficult and time-consuming job of selecting and coordinating the workshops was brought to a successful end by Yan Tang, and we were very glad to see that some of our earlier successful workshops (EI2N, META4eS, ISDE, INBAST, OntoContent) re-appeared in 2014, in some cases with a sixth or even eighth edition, and often in alliance with other older or newly emerging workshops. The new MSC workshop is an initiative of the same proposers of the erstwhile SOMOCO workshop. The Industry Case Studies Program, started in 2011 under the leadership of Hervé Panetto and OMG’s Richard Mark Soley, further gained momentum and visibility in its fourth edition this year.

The OTM registration format (“one workshop buys all”) actively intends to stimulate workshop audiences to productively mingle with each other and, optionally, with those of the main conferences. In particular EI2N continues to so create and exploit a visible synergy with CoopIS.

We were most happy to see that in 2014 the number of quality submissions for the OnTheMove Academy (OTMA) increased for the third consecutive year.

OTMA implements our unique, actively coached, formula to bring PhD students together, and aims to carry our “vision for the future” in research in the areas covered by OTM. Its 2014 edition was managed by a dedicated team of collaborators led by Peter Spyns and Maria-Ester Vidal, and of course inspired by the OTMA Dean, Erich Neuhold. In the OTM Academy, PhD research proposals are submitted by students for peer review; selected submissions and their approaches are to be presented by the students in front of a wider audience at the conference, and are independently and extensively analyzed and discussed in front of this audience by a panel of senior professors. One will readily appreciate the effort invested in this by the OTMA Faculty.

As the main conferences and the associated workshops all share the distributed aspects of modern computing systems, they experience the application pull created by the Internet and by the so-called Semantic Web. For ODBASE 2014, the focus continues to be the knowledge bases and methods required for enabling the use of formal semantics in Web-based databases and information systems. For CoopIS 2014, the focus as before was on the interaction of such technologies and methods with business process issues, such as occur in networked organizations and enterprises. These subject areas overlap in a scientifically natural fashion and many submissions in fact also treated an envisaged mutual impact among them. For our new core event C&TC 2014, the primary emphasis was now squarely put on the virtual and security aspects of Web-based computing in the broadest sense. As with the earlier OnTheMove editions, the organizers wanted to stimulate this cross-pollination by a program of famous keynote speakers around the chosen themes and shared by all OTM component events. We were proud to announce for this year:

- Domenico Saccà
- Ernesto Damiani
- Henk Sol
- Johann Eder

The general downturn in submissions observed in recent years for almost all conferences in computer science and IT this year finally also affected OnTheMove, but we were still fortunate to receive a total of 126 submissions for the three main conferences and 85 submissions in total for the workshops. Not only may we indeed again claim success in attracting a representative volume of scientific papers, many from the USA and Asia, but these numbers of course allow the respective Program Committees to again compose a high-quality cross-section of current research in the areas covered by OTM. Acceptance rates vary but the aim was to stay consistently at about one accepted paper for every two to three submitted, yet as always the rates are subordinated to professional peer assessment of proper scientific quality. As usual we have separated the proceedings into two volumes with their own titles, one for the main conferences and one for the workshops and posters, and we are again most grateful to the Springer LNCS team in Heidelberg for their professional support, suggestions, and meticulous collaboration in producing the files and indexes ready for downloading on the USB sticks.

VIII General Co-Chairs' Message for OnTheMove 2014

The reviewing process by the respective OTM Program Committees was performed to professional quality standards: Each paper in the main conferences was reviewed by at least three referees (four for most ODBASE papers), with arbitrated e-mail discussions in the case of strongly diverging evaluations. It may be worthwhile to emphasize once more that it is an explicit OnTheMove policy that all conference Program Committees and chairs make their selections in a completely sovereign manner, autonomous and independent from any OTM organizational considerations. As in recent years, proceedings in paper form are now only available to be ordered separately.

The general chairs are once more especially grateful to the many people directly or indirectly involved in the set-up of these federated conferences. Not everyone realizes the large number of persons that need to be involved, and the huge amount of work, commitment, and in the uncertain economic and funding climate of 2014 certainly also financial risk that is entailed by the organization of an event like OTM. Apart from the persons in their roles mentioned above, we therefore wish to thank in particular our main conference PC co-chairs:

- CoopIS 2014: Michele Missikoff, Lin Liu and Oscar Pastor
- ODBASE 2014: Alfredo Cuzzocrea and Timos Sellis
- C&TC 2014: Michele Bezzi and Henry Chan

And similarly we thank the 2014 OTMA and Workshops PC (co-)chairs (in order of appearance on the website): Alexis Aubry, Georg Weichhart, Ronald Giachetti, Michele Dassisti, Rafael Valencia García, Ricardo Colomo Palacios, Thomas Moser, Alok Mishra, Jürgen Münch, Deepti Mishra, Ioana Ciuciu, Anna Fensel, Fernando Ferri, Patrizia Grifoni, Arianna D'Ulizia, Maria Chiara Caschera, António Lucas Soares, Carla Sofia Pereira, Peter Spyns, Maria Esther Vidal, Anja Metzner, Erich J. Neuhold, and Alfred Holl.

All of them, together with their many PC members, performed a superb and professional job in managing the difficult yet essential process of peer review and selection of the best papers from the harvest of submissions. We all also owe our sincere gratitude to our supremely competent and experienced conference secretariat and technical support staff in Guadalajara and Brussels, respectively, Daniel Meersman and Jan Demey.

Two of the general co-chairs also thankfully acknowledge the academic freedom, logistic support, and facilities they enjoy from their respective institutions, Université de Lorraine CNRS, Nancy, France, and Latrobe University, Melbourne, Australia, without which such a project quite simply would not be feasible. We do hope that the results of this federated scientific enterprise contribute to your research and your place in the scientific network. We look forward to seeing you at next year's event!

September 2014

Robert Meersman
Hervé Panetto
Tharam Dillon

Organization

OTM (On The Move) is a federated event involving a series of major international conferences and workshops. These proceedings contain the papers presented at the OTM 2014 Federated conferences, consisting of CoopIS 2014 (Cooperative Information Systems) and ODBASE 2014 (Ontologies, Databases, and Applications of Semantics).

Executive Committee

General Co-chairs

Robert Meersman	TU Graz, Austria
Hervé Panetto	University of Lorraine, France
Tharam Dillon	La Trobe University, Melbourne, Australia

OnTheMove Academy Dean

Erich Neuhold	University of Vienna, Austria
---------------	-------------------------------

Industry Case Studies Program Chairs

Hervé Panetto	University of Lorraine, France
---------------	--------------------------------

CoopIS 2014 PC Co-Chairs

Michele Missikoff	Università Politecnica delle Marche and CNR, Italy
Lin Liu	Tsinghua University, China
Oscar Pastor	Universidad Politècnica de Valencia, Spain

ODBASE 2014 PC Co-Chairs

Alfredo Cuzzocrea	ICAR-CNR and University of Calabria, Italy
Timos Sellis	RMIT University, VIC, Australia

Logistics Team

Daniel Meersman
Jan Demey

CoopIS 2014 Program Committee

Akhil Kumar	Lijie Wen
Alfredo Cuzzocrea	Manfred Reichert
Antonio Ruiz Cortés	Marco Aiello
Asuman Dogac	Maria Esther Vidal
Barbara Weber	Maristella Matera
Barbara Pernici	Martin Zelm
Benjamin Knoke	Martine Collard
Carlo Combi	Massimo Canducci
Carlos Cetina	Massimo Mecella
Claudia Diamantini	Mathias Weske
Djamal Benslimane	Min Zhou
Epaminondas Kapetanios	Mohand-Said Hacid
Eric Yu	Nacer Boudjlida
Fei He	Paolo Giorgini
Fenglin Li	Ralf Schenkel
François B. Vernadat	Rania Khalaf
Francesco Taglino	Rik Eshuis
Gash Bhullar	Sanjay K. Madria
Gerald Oster	Schahram Dustdar
Giancarlo Guizzardi	Selmin Nurcan
Heinrich Mayr	Shazia Sadiq
Hiroyuki Kitagawa	Simon Schlosser
Hongji Yang	Stefan Jablonski
Jan Mendling	Susan Urban
Jianwen Su	Ted Goranson
Johann Eder	Tiziana Margaria
John Miller	Vicente Pelechano
Jolita Ralyté	Xavier Franch
Joonsoo Bae	Xiaojun Ye
Jose Luis Garrido	Xiaoping Sun
Julius Köpke	Zohra Bellahsene
Lakshmish Ramaswamy	

ODBASE 2014 Program Committee

Alicia Diaz	Dimitris Plexousakis
Amit Joshi	Eduardo Mena
Andrea Cali	Frederick Maier
Christian Kop	Geert Poels
Christophe Gueret	Guilin Qi
Christophe Debruyne	Harry Halpin
Davor Meersman	Ivana Podnar Zarko

Josiane Xavier Parreira
Kai-Uwe Sattler
Ladjel Bellatreche
Lei Zou
Manolis Koubarakis
Marie-Aude Aufaure
Maurizio Lenzerini
Michael Schrefl
Mohand-Said Hacid
Paea LePendu
Payam Barnaghi
Philippe Cudré-Mauroux
Prasad Deshpande
Prateek Jain

Rajasekar Krishnamurthy
Satya Sahoo
Sergio Greco
Sonia Bergamaschi
Srinath Srinivasa
Steffen Lamparter
Sunil Choenni
Takahiro Hara
Walid Gaaloul
Yanghua Xiao
Yannis Stavrakas
Yuan An
Zohra Bellahsene

OnTheMove 2014 Keynotes

Mining and Posting Big Data on the Web to Add Value to Information Contents

Domenico Saccà

University of Calabria, Italy

Short Bio

Domenico Saccà (<http://sacca.deis.unical.it>) is full professor of Computer Engineering at the University of Calabria since 1987 and he is presently chairing the School of Computer Engineering at that university.

Since 2009 he is also President of the Computing and Telecommunication Competence Center ICT-SUD, operating in five regions of Southern Italy: Calabria, Campania, Puglia, Sardegna and Sicilia.

From January 2002 to January 2009, he was Director of the CNR (the Italian National Research Council) Research Institute ICAR (Institute for High Performance Computing and Networking), located in Rende (CS) and branches in Naples and Palermo. Previously, from 1995 on, he was director of the CNR ISI (Institute on System and Computer Sciences).

In the past he was visiting scientist at IBM Laboratory of San Jose, at the Computer Science Department of UCLA and at the ICSI Institute of Berkeley; moreover, he was scientific consultant of MCC, Austin and manager of the Research Division of CRAI (a research institute in southern Italy).

His current research interests focus on advanced issues of databases such as: data and process mining, inverse data mining, data warehousing and OLAP on distributed platforms, compressed representation of datacubes, database query languages. Recently he has started some research activities on cyber security topics such as protection of systems, services and end users as well as on privacy issues.

His list of publications contains more than 200 papers on journals (including Journal of the ACM, SIAM Journal on Computing, ACM Transactions on Database Systems, Theoretical Computer Science, IEEE Transactions on Software Engineering, IEEE Transactions on Knowledge and Data Engineering, VLDB Journal, Information Systems, ACM Transactions on Knowledge Discovery from Data) and on proceedings of international conferences.

He has been member of the program committees of several international conferences and director of international schools and seminars.

Talk

“Mining and Posting Big Data on the Web to add value to information contents”

A multi-dimensional view of WEB Search is first presented to substantiate the speaker's position: a powerful search needs enriched information, which is to be published with additional discovered attributes (hidden dimensions).

Some classical DB theory tools (e.g., Data exchange and DATALOG) can be re-considered as powerful frameworks for creating multi-dimensional views for their flexible search without having to fight with SQL syntax.

Data mining can be used not only for discovering knowledge from WEB data but also for adding value to data to be published on the WEB.

An ambitious goal is to combine the strengths of both SQL and NOSQL approaches: the structural approach of SQL to enrich documents and the “unstructured” approach of NOSQL to free the search from join harassment and to enable scalable performance.

Cloud Assurance: The Notion and the Issues

Ernesto Damiani

Università degli Studi di Milano, Italy

Short Bio

Ernesto Damiani is a full professor at the Computer Science Department of Università degli Studi di Milano, Italy, the director of Secure Servcie-oriented Architectures (SESTAR) lab and the Head of the University's Ph.D. program in Computer Science. His areas of interest include Cloud and SOA security, semi-structured information processing, business process analysis and discovery. He has published several books and more than 300 papers and international patents. His work has appeared, among many others, in the IEEE Trans. on Knowledge and Data Engineering, the IEEE Trans. on Service Computing, the ACM Trans. on Information and System Security, the IEEE Trans. on Fuzzy Systems, the ACM Trans. on Information Systems and the ACM Trans. on Software Engineering and Methodology. He is a senior member of the IEEE and ACM Distinguished Scientist.

Ernesto Damiani leads/has led a number of international research projects: he was the Principal Investigator of the ASSERT4SOA project (STREP) on the security certification of SOA; has led the activity of SESAR research unit within SecureSCM (STREP), ARISTOTELE (IP), ASSERT4SOA (STREP), CUMULUS (STREP) and PRACTICE (IP) projects funded by the EC in the 7th Framework Program.

Ernesto has been an Associate Editor of the IEEE Trans. on Service-Oriented Computing since its inception, and is an Associate Editor of the IEEE Trans. on Fuzzy Systems. Also, Prof. Damiani is Editor in chief of the International Journal of Knowledge and Learning (Inderscience) and of the International Journal of Web Technology and Engineering (IJWTE).

Talk

“Cloud Assurance: The Notion and the Issues”

Generating and handling assurance information on the cloud is an open challenge, as conflicting requirements (e.g., transparency vs. privacy) are emerging and the size of data involved is huge. Still, managing assurance is of paramount importance for guaranteeing the desired security and dependability properties of cloud-based computations. In this talk, we first discuss the conceptual framework to represent monitoring and test-based assurance, grounding assurance-based service-level agreements (SLAs) and certification models for cloud-based services. Then, we focus on:

XVIII Cloud Assurance: The Notion and the Issues

- (i) the definition of security and dependability properties to be negotiated and certified on the cloud
- (ii) the types of evidence underlying them and the mechanisms for generating evidence
- (iii) the phases of the assurance artifacts life-cycle.

eHealthEnough or pHealth: Providing COLLAGEN for Innovations

Henk G. Sol

University of Groningen/Delft University of Technology, The Netherlands

Short Bio

Prof. dr. Henk G. Sol is over 40 years a driver of engaged scholarship in the field of information systems and decision enhancement. With his school of nearly 80 completed PhD dissertations supervised and some 25 PhD dissertations under way, he is a major contributor to building the foundations of design science research in management and information systems.

In addition, he is responsible for the graduation of over 700 Engineering MSc and MBA students. All dissertations are based on theory development applied to tackle issues that matter in practice and are relevant to both developing and developed countries.

As founding dean he was responsible for the establishment of the Faculty of Technology, Policy and Management at Delft University of Technology and of the Faculty of Economics and Business at the University of Groningen, the Netherlands.

Prof. Sol has organized numerous international conferences and workshops, of which the conference series on CRIS and Dynamic Modeling have been seminal. He has published widely in renowned journals, edited many books, and given many keynote presentations all over the world. He acted as a consultant to many governments and organizations worldwide. He was founding father of IFIP TC8, 8.1 and 8.3, AIS and many (inter)national doctoral consortia.

Henk G. Sol serves currently in various academic and professional roles: President of the Supervisory Board of Groningen Airport Eelde NV, Eelde; Director of Sol Information Management BV, Haren; Chairperson of the Board of Trustees, Uganda Technology and Management University, Kampala; Chairperson of Stichting PAO Informatica; Consulting Professor of PBLQ, The Hague; and Member of the Board of Trustees of the International Institute for Communication and Development, the Hague.

Talk

“eHealthEnough or pHealth: Providing COLLAGEN for Innovations”

Enhancing issues that matter is a major challenge in our ambient society, especially in the health domain: we have to navigate in the sea of information to deliver shared value. Agile, analytic, big, intelligent, smart, sustainable data describe the potential for decision enhancement. Processes have to be engineered accordingly to deliver shared value.

Many initiatives are taken around eHealth, but what is the impact of the many projects in terms of usefulness, usability and usage?

Studies on the effectiveness of eHealth may lead to the conclusion that eHealth is eNough. Many examples indicate that the patient is often neglected, that physical and decision processes are not looked into and that the technology is pushed, despite great development funds for e.g. eInclusion and Ambient Assisted Living.

Delivering shared value calls for: conversations to collectively identify locally relevant problems, governance to make political, administrative and business decisions about tackling such problems and engaged innovations to develop localized solutions.

The COLLAGEN (Collective Learning and Agile Governance Environment) approach provides a set of services for scoping, facilitation and enhancement of business processes, packed into decision apps and providing guidelines for conversational inquiry. The approach supports smart governance for business engineering and engaged innovations, and delivers shared value to resolve issues that matter in society, especially in health care and cure.

It is posited that innovations in health care demand pHealth for delivering shared value based on patient focus, personal intervention, process anchoring and participatory design.

Managing the Execution of Business Processes

Johann Eder

Alpen-Adria Universität Klagenfurt, Austria

Short Bio

Johann Eder is full professor for Information and Communication Systems in the Department of Informatics-Systems of the Alpen-Adria Universität Klagenfurt, Austria. From 2005-2013 he was Vice President of the Austrian Science Funds (FWF). He held positions at the Universities of Linz, Hamburg and Vienna and was visiting scholar at AT&T Shannon Labs.

The research interests of Johann Eder are databases, information systems and data management for medical research. He successfully directed many funded research projects on workflow management systems, temporal data warehousing, application interoperability, information systems modelling, information systems for medical research, etc.

Johann Eder has contributed to workflow systems and business process management for 2 decades, in particular in the area of workflow systems languages and architectures, exception handling, time management, and data management. His research led to the development of the commercial workflow management systems altavistaWorks and @enterprise.

He published more than 150 papers in international journals and conference proceedings. He served in numerous program committees for international conferences and as editor and referee for international journals. He acted as general chair and/or PC chair for CAiSE, ADBIS, BPM, CoopIS, and DAWAK conferences.

Talk

“Managing the Execution of Business Processes”

Managing the execution of business processes requires many decisions: whom to assign to a task, when to execute a task, how to deal with exceptions, etc. Some of these decisions can be automated according to some policies. Some cannot be automated but can be supported by process technologies.

In this talk we will analyze which decisions have to be taken at the run-time of business processes by process participants and by process managers and discuss how these decisions can be supported by components of workflow management systems, in particular time management systems, resource management systems, scheduling systems, exception handling systems, and process warehousing.

Table of Contents

OnTheMove 2014 Keynotes

Mining and Posting Big Data on the Web to Add value to Information Contents	XV
<i>Domenico Saccà</i>	
Cloud Assurance: The Notion and the Issues	XVII
<i>Ernesto Damiani</i>	
eHealtheNough or pHealth: Providing COLLAGEN for Ennovations	XIX
<i>Henk G. Sol</i>	
Managing the Execution of Business Processes	XXI
<i>Johann Eder</i>	

Cooperative Information Systems (CoopIS) 2014

CoopIS 2014 PC Co-Chairs Message

Process Design and Modeling

Decomposing Alignment-Based Conformance Checking of Data-Aware Process Models	3
<i>Massimiliano de Leoni, Jorge Munoz-Gama, Josep Carmona, and Wil M.P. van der Aalst</i>	
A Pattern Approach to Conquer the Data Complexity in Simulation Workflow Design	21
<i>Peter Reimann, Holger Schwarz, and Bernhard Mitschang</i>	
Augmenting and Assisting Model Elicitation Tasks with 3D Virtual World Context Metadata	39
<i>Ross Brown, Stefanie Rinderle-Ma, Simone Kriglstein, and Sonja Kabicher-Fuchs</i>	

Deployment of Service-Based Processes in the Cloud Using Petri Net Decomposition.....	57
<i>Sami Yangui, Kais Klai, and Samir Tata</i>	

Process Enactment

Log-Based Understanding of Business Processes through Temporal Logic Query Checking	75
<i>Margus Räim, Claudio Di Ciccio, Fabrizio Maria Maggi, Massimo Mecella, and Jan Mendling</i>	
Approach and Refinement Strategies for Flexible Choreography Enactment	93
<i>Andreas Weiß, Santiago Gómez Sáez, Michael Hahn, and Dimka Karastoyanova</i>	
Business Process Fragments Behavioral Merge	112
<i>Mohamed Anis Zemni, Nejib Ben Hadj-Alouane, and Amel Mammar</i>	

Monitoring and Quality Assessment

Provenance-Based Quality Assessment and Inference in Data-Centric Workflow Executions	130
<i>Clément Caron, Bernd Amann, Camelia Constantin, Patrick Giroux, and André Santanchè</i>	
Collaborative Building of an Ontology of Key Performance Indicators ...	148
<i>Claudia Diamantini, Laura Genga, Domenico Potena, and Emanuele Storti</i>	
Aligning Monitoring and Compliance Requirements in Evolving Business Networks	166
<i>Marco Comuzzi</i>	

Managing Similarity

TAGER: Transition-Labeled Graph Edit Distance Similarity Measure on Process Models	184
<i>Zixuan Wang, Lijie Wen, Jianmin Wang, and Shuhao Wang</i>	
CFS: A Behavioral Similarity Algorithm for Process Models Based on Complete Firing Sequences	202
<i>Zihe Dong, Lijie Wen, Haowei Huang, and Jianmin Wang</i>	
Efficient Behavioral-Difference Detection between Business Process Models	220
<i>Zhiqiang Yan, Yuquan Wang, Lijie Wen, and Jianmin Wang</i>	

Compliance Checking of Data-Aware and Resource-Aware Compliance Requirements	237
<i>Elham Ramezani Taghiabadi, Vladimir Gromov, Dirk Fahland, and Wil M.P. van der Aalst</i>	

Software Services

RelBOSS: A Relationship-Aware Access Control Framework for Software Services	258
<i>A.S.M. Kayes, Jun Han, Alan Colman, and Md. Saiful Islam</i>	
A QoS-Aware, Trust-Based Aggregation Model for Grid Federations	277
<i>Antonello Comi, Lidia Fotia, Fabrizio Messina, Domenico Rosaci, and Giuseppe M.L. Sarnè</i>	
Towards a Formal Specification of SLAs with Compensations	295
<i>Carlos Müller, Antonio M. Gutiérrez, Octavio Martín-Díaz, Manuel Resinas, Pablo Fernández, and Antonio Ruiz-Cortés</i>	
Steady Network Service Design for Seamless Inter-cloud VM Migration	313
<i>Nihed Bahria El Asghar, Omar Cherkaoui, Mounir Frikha, and Sami Tabbane</i>	

Improving Alignment

Prognosing the Compliance of Declarative Business Processes Using Event Trace Robustness	327
<i>Maria Teresa Gómez-López, Luisa Parody, Rafael M. Gasca, and Stefanie Rinderle-Ma</i>	
Event-Based Real-Time Decomposed Conformance Analysis	345
<i>Seppe K.L.M. vanden Broucke, Jorge Muñoz-Gama, Josep Carmona, Bart Baesens, and Jan Vanthienen</i>	
Capitalizing the Designers' Experience for Improving Web API Selection	364
<i>Devis Bianchini, Valeria De Antonellis, and Michele Melchiori</i>	

Collaboration Systems and Applications

Software Support Requirements for Awareness in Collaborative Modeling	382
<i>Michel Dirix, Xavier Le Pallec, and Alexis Muller</i>	

Trusted Dynamic Storage for Dunbar-Based P2P Online Social Networks	400
<i>Marco Conti, Andrea De Salve, Barbara Guidi, Francesco Pitto, and Laura Ricci</i>	
A Cooperative Information System for Managing Traffic Incidents with the PAUSETA Protocol.....	418
<i>Miguel Prades-Farrón, Luis A. García, and Vicente R. Tomás</i>	

Short Papers

Mining Business Process Deviance: A Quest for Accuracy	436
<i>Hoang Nguyen, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Suriadi Suriadi</i>	
Multi-paradigm Process Mining: Retrieving Better Models by Combining Rules and Sequences	446
<i>Johannes De Smedt, Jochen De Weerdt, and Jan Vanthienen</i>	

Ontologies, DataBases, and Applications of Semantics (ODBASE) 2014

ODBASE 2014 PC Co-Chairs Message

Ontology Querying Methodologies and Paradigms

Fuzzy XPath Queries in XQuery	457
<i>Jesús M. Almendros-Jiménez, Alejandro Luna, and Ginés Moreno</i>	
Flexible Querying for SPARQL	473
<i>Andrea Calì, Riccardo Frosini, Alexandra Poulovassilis, and Peter T. Wood</i>	

How Good Is Your SPARQL Endpoint? A QoS-Aware SPARQL Endpoint Monitoring and Data Source Selection Mechanism for Federated SPARQL Queries	491
<i>Muhammad Intizar Ali and Alessandra Mileo</i>	

Embedding OWL Querying and Reasoning into XQuery	509
<i>Jesús M. Almendros-Jiménez</i>	

Ontology Support for Web, XML and RDF Data Processing and Retrieval

Deriving Folksonomies for Improving Web API Search	517
<i>Devis Bianchini</i>	

Adaptive Similarity of XML Data	535
<i>Eva Jilková, Marek Polák, and Irena Holubová</i>	

FAGI: A Framework for Fusing Geospatial RDF Data	553
<i>Giorgos Giannopoulos, Dimitrios Skoutas, Thomas Maroulis, Nikos Karagiannakis, and Spiros Athanasiou</i>	

Knowledge Bases Querying and Retrieval

Online Reasoning for Ontology-Based Error Detection in Text	562
<i>Fernando Gutierrez, Dejing Dou, Stephen Fickas, and Gina Griffiths</i>	

Making Metaquerying Practical for $Hi(DL-Lite_{\mathcal{R}})$ Knowledge Bases	580
<i>Maurizio Lenzerini, Lorenzo Lepore, and Antonella Poggi</i>	

Knowledge-Based Compliance Checking of Business Processes	597
<i>Ildikó Szabó and Krisztián Varga</i>	

Improved Automatic Maturity Assessment of Wikipedia Medical Articles (Short Paper)	612
<i>Emanuel Marzini, Angelo Spognardi, Ilaria Matteucci, Paolo Mori, Marinella Petrocchi, and Riccardo Conti</i>	

Social Network and Collaborative Methodologies

Integrity Management in a Trusted Utilitarian Data Exchange Platform	623
<i>Sweety Agrawal, Chinmay Jog, and Srinath Srinivasa</i>	

A Model to Support Multi-Social-Network Applications	639
<i>Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera</i>	

A Method for Detecting Behavior-Based User Profiles in Collaborative Ontology Engineering	657
<i>Sven Van Laere, Ronald Buyl, and Marc Nyssen</i>	

Defining and Investigating the Scope of Users and Hashtags in Twitter (Short Paper)	674
<i>Daniel Leggio, Giuseppe Marra, and Domenico Ursino</i>	

Ontology-Assisted Event and Stream Processing

Constructing Event Processing Systems of Layered and Heterogeneous Events with SPARQL	682
<i>Mikko Rinne and Esko Nuutila</i>	

XXVIII Table of Contents

On Efficient Processing of Linked Stream Data	700
<i>Omran Saleh and Kai-Uwe Sattler</i>	
Applying Semantics to Optimize End-User Services in Telecommunication Networks	718
<i>Liam Fallon, John Keeney, and Declan O'Sullivan</i>	

Ontology-Assisted Warehousing Approaches

An Ontology-Based Data Exploration Tool for Key Performance Indicators	727
<i>Claudia Diamantini, Domenico Potena, Emanuele Storti, and Haotian Zhang</i>	
Arabic-English Domain Terminology Extraction from Aligned Corpora	745
<i>Wiem Lahbib, Ibrahim Bounhas, and Bilel Elayeb</i>	

Towards a Configurable Database Design: A Case of Semantic Data Warehouses	760
<i>Selma Khouri and Ladjel Bellatreche</i>	

Ontology-Based Data Representation and Management in Emerging Domains

Describing Research Data: A Case Study for Archaeology	768
<i>Nicola Aloia, Christos Papatheodorou, Dimitris Gavriliis, Franca Debole, and Carlo Meghini</i>	
Enriching Semantically Web Service Descriptions	776
<i>Maricela Bravo, José Rodríguez, and Alejandro Reyes</i>	
Parameterized Algorithms for Matching and Ranking Web Services (Short Paper)	784
<i>Fatma Ezzahra Gmati, Nadia Yacoubi-Ayadi, and Salem Chakhar</i>	
Arabic Domain Terminology Extraction: A Literature Review (Short Paper)	792
<i>Ibrahim Bounhas, Wiem Lahbib, and Bilel Elayeb</i>	

Erratum

Flexible Querying for SPARQL	E1
<i>Andrea Calì, Riccardo Frosini, Alexandra Poulovassilis, and Peter T. Wood</i>	

Author Index	801
---------------------------	-----

CoopIS 2014 PC Co-chairs Message

CoopIS represents one of the most qualified and well-established international conferences positioned in a scientific area focusing on the engineering of information systems with a strong characterization on “Cooperation”. It is then characterized by the cross fertilization of several disciplines, from information systems to service oriented architectures, from business processes and workflows to monitoring and application systems assessment.

CoopIS 2014, being part of the Federated Conferences “On The Move”, this year has attracted 68 submissions that have been carefully evaluated by a massive review process that involved more than 80 PC Members who have produced more than 250 review reports. After the evaluation process we were able to select 24 full papers (with a selection rate of 35%), plus 2 short papers and 2 posters.

CoopIS 2014 has continued attracting important research results, amongst which the most represented research area is related to business process modeling and management covering various aspects, from the design phase and the process enactment (7 full papers) to behavioral similarity analysis (4 full papers). Connected to the latter, a group of 3 papers are concerned with process alignment. The second most popular topic concerns services and service-oriented computing (7 full papers), followed by methods for monitoring and assessment of processes (3 full papers). Other topics concern event-management and cooperative applications. Also the 3 short paper are concerned with process analysis. We believe that such a rich scientific program will provide an inspiring showcase of top researches going on in the mentioned areas.

Once more we wish to recall that the rich scientific program offered by CoopIS 2014 has been possible primarily thanks to the authors and their high quality papers, by the dedicated work of the Program Committee, and, last but not least, by the intense work of the OTM organization, in particular of Daniel Meersman and Jan Demey, with the wise supervision of Robert Meersman. We wish to recall that the scientific program is complemented by the Industrial Program chaired by Hervé Panetto.

July 2014

Michele Missikoff
Lin Liu
Oscar Pastor

Decomposing Alignment-Based Conformance Checking of Data-Aware Process Models

Massimiliano de Leoni^{1,*}, Jorge Muñoz-Gama^{2,**},
Josep Carmona², and Wil M.P. van der Aalst¹

¹ Eindhoven University of Technology, Eindhoven, The Netherlands

² Universitat Politècnica de Catalunya, Barcelona, Spain

m.d.leoni@tue.nl, {jmunoz, jcarmona}@cs.upc.edu,
w.m.p.v.d.aalst@tue.nl

Abstract. Process mining techniques relate observed behavior to modeled behavior, e.g., the automatic discovery of a Petri net based on an event log. Process mining is not limited to process discovery and also includes conformance checking. Conformance checking techniques are used for evaluating the quality of discovered process models and to diagnose deviations from some normative model (e.g., to check compliance). Existing conformance checking approaches typically focus on the control-flow, thus being unable to diagnose deviations concerning data. This paper proposes a technique to check the conformance of data-aware process models. We use so-called *Petri nets with Data* to model data variables, guards, and read/write actions. Data-aware conformance checking problem may be very time consuming and sometimes even intractable when there are many transitions and data variables. Therefore, we propose a technique to decompose large data-aware conformance checking problems into smaller problems that can be solved more efficiently. We provide a general correctness result showing that decomposition does not influence the outcome of conformance checking. The approach is supported through ProM plug-ins and experimental results show significant performance improvements. Experiments have also been conducted with a real-life case study, thus showing that the approach is also relevant in real business settings.

Keywords: Process Mining, Conformance Checking, Divide-and-Conquer Techniques, Multi-Perspective Process Modelling.

1 Introduction

Nowadays, most organizations document and analyze their processes in some form, and with it, the practical relevance of process mining is increasing as more and more event data becomes available. Process mining techniques aim to discover, monitor and improve real processes by extracting knowledge from event logs. The two most prominent process mining tasks are: (i) *process discovery*: learning a process model from example behavior recorded in an event log, and (ii) *conformance checking*: diagnosing

* When conducting most of this research work, Dr. de Leoni was also affiliated with University of Padua and financially supported by the Eurostars - Eureka project PROMPT (E!6696).

** Supported by FPU Grant (AP2009-4959) and project FORMALISM (TIN-2007-66523).

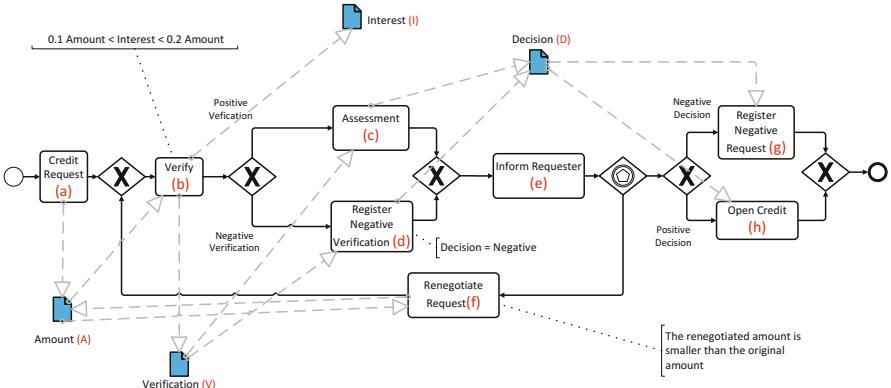


Fig. 1. Example of a (simplified) process to request loans. The dotted arcs going from a transition to a variable denote write operations; the arcs towards a transition denote read operations, i.e. the transition requires accessing the current variables' value. In the paper, each transition is abbreviated into a lower-case letter (e.g. *a*) and each variable is represented as a upper-case letter (e.g. *A*). The abbreviations are shown in brackets after the name of the transitions or variable names.

and quantifying discrepancies between observed behavior and modeled behavior [1]. Models that faithfully conform the reality are necessary to obtain trustful analysis and simulation results, for certification and regulation purposes, or simply to gain insight into the process.

Most of the work done in conformance checking in the literature focuses on the control-flow of the underlying process, i.e. the ordering of activities. There are various approaches to compute the fraction of events or traces in the log that can be replayed by the model [2,3].

In a data-aware process model, each case, i.e. a process instance, is characterized by its case variables. Paths taken during the execution may be governed by guards and conditions defined over such variables. A process model specifies the set of variables and their possible values, guards, and write/read actions. Since existing conformance checking techniques typically completely abstract from data, resources, and time, many deviations remain undetected. Therefore, the event log may record executions of process instances that appear fully conforming, even when it is not the case. Rigorous analysis of the data perspective is needed to reveal such deviations.

Let us consider the process that is modeled as BPMN diagram in Figure 1. It models the handling of loans requests from customers. It is deliberately oversimplified to be able to explain the concepts more easily. The process starts with a credit request where the requestor provides some documents to demonstrate the capability of paying the loan back. These documents are verified and the interest amount is also computed. If the verification step is negative, a negative decision is made, the requestor is informed and, finally, the negative outcome of the request is stored in the system. If verification is positive, an assessment is made to take a final decision. Independently of the assessment's decision, the requestor is informed. Moreover, even if the verification is negative, the requestor can renegotiate the loan (e.g. to have lower interests) by providing further documents or by asking for a smaller amount. In this case, the verification-assessment

part is repeated. If both the decision and verification are positive and the requestor is not willing to renegotiate, the credit is opened. Let us consider the following trace:¹

$$\sigma_{ex} = \langle (\mathbf{a}, \emptyset, \{(A, 4000)\}), (\mathbf{b}, \{(A, 4000)\}, \{(I, 450), (V, \text{false})\}), (\mathbf{c}, \{(V, \text{false})\}, \{(D, \text{true})\}), (\mathbf{e}, \emptyset, \emptyset), (\mathbf{f}, \{(A, 4000)\}, \{(A, 5000)\}), (\mathbf{b}, \{(A, 5000)\}, \{(I, 450), (V, \text{false})\}), (\mathbf{d}, \{(V, \text{false})\}, \{(D, \text{false})\}), (\mathbf{e}, \emptyset, \emptyset), (\mathbf{h}, \{(D, \text{true})\}, \emptyset) \rangle$$

Seen from a control-flow perspective only (i.e. only considering the activities' ordering), the trace seems to be fully conforming. Nonetheless, a number of deviations can be noticed if the data perspective is considered. First of all, if activity c is executed, previously activity b could not have resulted in a negative verification, i.e. V is set to **false**. Second, activity f cannot write value 5000 to variable A , as this new value is larger than the previous value, i.e. 4000. Furthermore, if the decision and verification are both negative, i.e. both V and D are set to **false**, then h cannot be executed at the end.

The identification of non-conforming traces clearly has value in itself. Nonetheless, organizations are often interested in explanations that can steer measures to improve the quality of the process. *Alignments* aim to support more refined conformance checking. An alignment aligns a case in the event log with an execution path of the process model as good as possible. If the case deviates from the model, then it is not possible to perfectly align with the model and a best matching scenario is selected. Note that for the same deviation, multiple explanations can be given. For instance, the problem that h was executed when it was not supposed to happen can be explained in two ways: (1) h should not have occurred because V and D are both set to **false** (“control-flow is wrong”) and (2) V and D should both have been set to **true** because h occurs (“data-flow is wrong”). In order to decide for the most reasonable explanation, costs are assigned to deviations and we aim to find the explanation with the lowest cost. For instance, if assigning a wrong value to V and D is less severe than executing h wrongly, the second explanation is preferred. The seminal work in [3] only considers alignments in the control-flow part, thus ignoring the data-perspective aspect of conformance.

As we detail in Section 2.4, finding an alignment of an event log and a data-aware process model is undecidable in the general case. However, to make the problem decidable, works [4,5] put forward the limitation that guards need to be linear (in)equations. Readers are also referred to them for a state-of-the-art analysis of data-aware conformance checking. These works also show that, even with that limitation, the problem of finding an alignment of an event log can become intractable since the problem's complexity is exponential on the size of the model, i.e. the number of activities and data variables. In this paper, while keeping the limitations mentioned above, we aim to speed up the computation of alignments by using a divide-and-conquer approach. The data-aware process model is split into smaller partly overlapping model fragments. For each model fragment a sublog is created by projecting the initial event log onto the activities used in the fragment. Given the exponential nature of conformance checking, this may

¹ Notation (\mathbf{act}, r, w) is used to denote the occurrence of activity \mathbf{act} that writes and reads variables according to functions w and r , e.g., $(\mathbf{b}, \{(A, 4000)\}, \{(I, 450), (V, \text{false})\})$ is an event corresponding to the occurrence of activity \mathbf{b} while reading value 4000 for variable A and writing values 450 and **false** to variables I and V respectively. $(\mathbf{e}, \emptyset, \emptyset)$ corresponds to the occurrence of activity \mathbf{e} without reading/writing any variables.

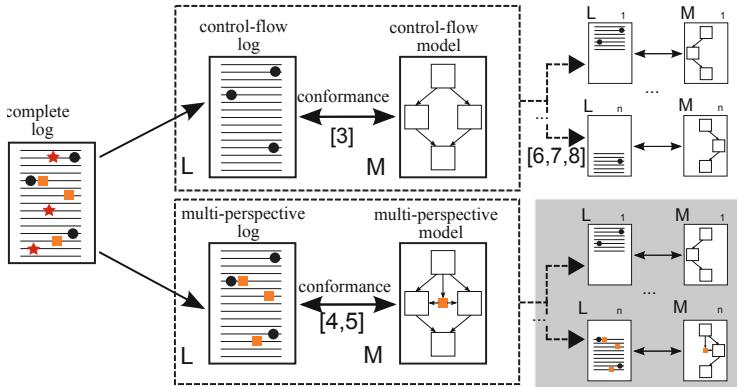


Fig. 2. Positioning the contribution of this paper with respect to the state of the art: the gray area identifies the novelty of the proposed technique

significantly reduce the computation time. If the decomposition is done properly, then any trace that fits into the overall model also fits all of the smaller model fragments and vice versa. Figure 2 positions the contribution of this paper with respect to the state-of-the-art alignment-based techniques. The top part identifies the alignment-based conformance checking that considers the control flow, only. The bottom part refers to the alignment-based conformance checking techniques that account for data, as well. Regarding the control-flow only, several approaches have been proposed to decompose process mining problems, both for discovery and conformance checking. As described in [6], it is possible to decompose process mining problems in a variety of ways. Special cases of this more general theory are passages [7] and SESE-based decomposition [8]. However, these approaches are limited to control-flow. Indeed, some techniques exist that also consider the data aspects (i.e. [4,5]) but without exploiting the possibility of decomposing the data-aware model. In this paper, we extend the control-flow approaches mentioned above to also take data into account, which coincides with the gray area in Figure 2. Finally, the work in [9] (and similar) for data-aware conformance on declarative models is orthogonal to the contributions listed in Figure 2, that are focused on procedural models.

The decomposed data-aware conformance checking approach presented in this paper has been implemented as plug-ins for the ProM framework. We conducted experiments related to a real-life case study as well as with several synthetic event logs. Experimental results show that data-aware decomposition may indeed be used to significantly reduce the time needed for conformance checking and that the problem is practically relevant since models of real processes can actually be decomposed.

Preliminaries are presented in Section 2. Section 3 introduces our approach for data-aware decomposition. Section 4 describes different algorithms for instantiating the general results presented in Section 3. Section 5 reports on experimental results. Section 6 concludes the paper.

2 Preliminaries

2.1 System Nets

Petri nets and their semantics are defined as usual: a Petri net is a tuple (P, T, F) with P the set of places, T the set of transitions, $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ the flow relation. A place p is an input place of a transition t iff $(p, t) \in F$; similarly, p is an output place of t iff $(t, p) \in F$. The marking of a Petri net is a multiset of tokens, i.e., $M \in \mathbb{B}(P)$. For some multiset $M \in \mathbb{B}(P)$, $M(p)$ denotes the number of times element p appears in M . The standard set operators can be extended to multisets, $M_1 \uplus M_2$ is the union of two multisets.

Firing a transition t in a marking M consumes one token from each of its input places and produces one token in each of its output places. Furthermore, transition t is enabled and may fire in M if there are enough tokens in its input places for the consumptions to be possible, i.e. iff for each input place s of t , $M(s) \geq 1$. Some of the transitions corresponds to piece of work in the process; each of those transitions are associated with a label that indicates the activity that it represents.

Definition 1 (Labeled Petri net). A labeled Petri net $PN = (P, T, F, l)$ is a Petri net (P, T, F) with labeling function $l \in T \not\rightarrow \mathcal{U}_A$ where \mathcal{U}_A is some universe of activity labels.²

Transitions without a label are invisible transitions, also known as τ -transitions. They are introduced for routing purposes but they do not represent actual pieces of work. As such, their execution is not recorded in the event logs.

Definition 2 (System Net). A system net $SN = (PN, M_{init}, M_{final})$ is a triplet where $PN = (P, T, F, l)$ is a labeled Petri net, $M_{init} \in \mathbb{B}(P)$ is the initial marking, and $M_{final} \in \mathbb{B}(P)$ is the final marking. \mathcal{U}_{SN} is the universe of system nets.

Definition 3 (System Net Notations). Let $SN = (PN, M_{init}, M_{final}) \in \mathcal{U}_{SN}$ be a system net with $PN = (P, T, F, l)$.

- $T_v(SN) = \text{dom}(l)$ is the set of visible transitions in SN ,
- $A_v(SN) = \text{rng}(l)$ is the set of corresponding observable activities in SN ,
- $T_v^u(SN) = \{t \in T_v(SN) \mid \forall t' \in T_v(SN) l(t) = l(t') \Rightarrow t = t'\}$ is the set of unique visible transitions in SN (i.e., there are no other transitions having the same visible label), and
- $A_v^u(SN) = \{l(t) \mid t \in T_v^u(SN)\}$ is the set of corresponding unique observable activities in SN .

In the remainder, for the formal definitions and proved theorems in Section 3, we need to introduce the concept of union of two nets. For this, we need to merge labeling functions. For any two partial functions $f_1 \in X_1 \not\rightarrow Y_1$ and $f_2 \in X_2 \not\rightarrow Y_2$: $f_3 = f_1 \oplus f_2$ is the union of the two functions. $f_3 \in (X_1 \cup X_2) \not\rightarrow (Y_1 \cup Y_2)$, $\text{dom}(f_3) = \text{dom}(f_1) \cup \text{dom}(f_2)$, $f_3(x) = f_2(x)$ if $x \in \text{dom}(f_2)$, and $f_3(x) = f_1(x)$ if $x \in \text{dom}(f_1) \setminus \text{dom}(f_2)$.

² Symbol $\not\rightarrow$ is used to denote partial functions.

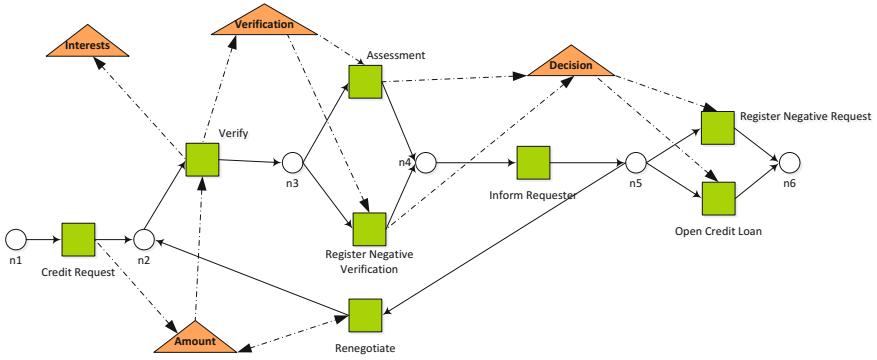


Fig. 3. Pictorial representation of a Petri net with Data that models the process earlier described in terms of BPMN diagram (cf. Figure 1). Places, transitions and variables are represented as circles, rectangles and triangles, respectively. The dotted arcs going from a transition to a variable denote the writing operations; the reverse arcs denote the read operations, i.e. the transition requires accessing the current variables' value.

Definition 4 (Union of Nets). Let $SN^1 = (N^1, M_{init}^1, M_{final}^1) \in \mathcal{U}_{SN}$ with $N^1 = (P^1, T^1, F^1, l^1)$ and $SN^2 = (N^2, M_{init}^2, M_{final}^2) \in \mathcal{U}_{SN}$ with $N^2 = (P^2, T^2, F^2, l^2)$ be two system nets.

- $l^3 = l^1 \oplus l^2$ is the union of l_1 and l_2 ,
- $N^1 \cup N^2 = (P^1 \cup P^2, T^1 \cup T^2, F^1 \cup F^2, l^3)$ is the union of N^1 and N^2 , and
- $SN^1 \cup SN^2 = (N^1 \cup N^2, M_{init}^1 \uplus M_{init}^2, M_{final}^1 \uplus M_{final}^2)$ is the union of system nets SN^1 and SN^2 .

2.2 Petri Nets with Data

A Petri net with Data is a Petri net with any number of variables (see Definitions 5 and 6 below). Petri nets with data can be seen as an abstracted version of high-level/colored Petri nets [10]. Colored Petri nets are extremely rich in expressiveness; however, many aspects are unimportant in our setting. Petri nets with data provide precisely the information needed for conformance checking of data-aware models and logs.

Definition 5 (Variables and Values). \mathcal{U}_{VN} is the universe of variable names. \mathcal{U}_{VV} is the universe of values. $\mathcal{U}_{VM} = \mathcal{U}_{VN} \not\rightarrow \mathcal{U}_{VV}$ is the universe of variable mappings.

In this type of nets, transitions may read from and/or write to variables. Moreover, transitions are associated with guards over these variables, which define when these they can fire. A guard can be any formula over the process variables using relational operators ($<$, $>$, $=$) as well as logical operators such as conjunction (\wedge), disjunction (\vee), and negation (\neg). A variable v appear as v_r or v_w , denoting the values read and written by the transition for v . We denote with $Formulas(V)$ the universe of such formulas defined over a set V of variables. In the remainder, given a set $V \subset \mathcal{U}_{VN}$ of variable names, we denote $V_R = \{v_r : v \in V\}$ and $V_W = \{v_w : v \in V\}$.

Formally, a *Petri net with Data* (DPN) is defined as follows:

Definition 6 (Petri net with Data). A *Petri net with Data DPN* = $(SN, V, val, init, read, write, guard)$ consists of

- a system net $SN = (PN, M_{init}, M_{final})$ with $PN = (P, T, F, l)$,
- a set $V \subseteq \mathcal{U}_{VN}$ of data variables,
- a function $val \in V \rightarrow \mathcal{P}(\mathcal{U}_{VV})$ that defines the values admissible for each variable, i.e., $val(v)$ is the set of values that variable v can have,³
- a function $init \in V \rightarrow \mathcal{U}_{VV}$ that defines the initial value for each variable v such that $init(v) \in val(v)$ (initial values are admissible),
- a read function $read \in T \rightarrow \mathcal{P}(V)$ that labels each transition with the set of variables that it reads,
- a write function $write \in T \rightarrow \mathcal{P}(V)$ that labels each transition with the set of variables that it writes,
- a guard function $guard \in T \rightarrow Formulas(V_W \cup V_R)$ that associates a guard with each transition such that, for any $t \in T$ and for any $v \in V$, if v_r appears in $guard(t)$ then $v \in read(t)$ and if v_w appears in $guard(t)$ then $v \in write(t)$.

\mathcal{U}_{DPN} is the universe of Petri nets with data.

The notion of bindings is essential for the remainder. A *binding* is a triplet (t, r, w) describing the execution of transition t while reading values r and writing values w . A binding is valid if:

1. $r \in read(t) \rightarrow \mathcal{U}_{VV}$ and $w \in write(t) \rightarrow \mathcal{U}_{VV}$
2. for any $v \in read(t)$: $r(v) \in val(v)$, i.e., all values read should be admissible,
3. for any $v \in write(t)$: $w(v) \in val(v)$, i.e., all values written should be admissible.
4. Guard $guard(t)$ evaluate true.

More specifically, let us introduce variable assignment $\chi_b : (V_R \cup V_W) \not\rightarrow \mathcal{U}_{VV}$ which is defined as follows: for any $v \in read(t)$, $\chi(v_r) = r(v)$ and, for any $v \in write(t)$, $\chi(v_w) = w(v)$. A binding (t, r, w) makes $guard(t)$ evaluate true if the evaluation of $guard(t)$ wrt. χ_b returns true.

A marking (M, s) of a Petri net with Data DPN has two components: $M \in \mathbb{B}(P)$ is the *control-flow marking* and $s \in \mathcal{U}_{VM}$ with $dom(s) = V$ and $s(v) \in val(v)$ for all $v \in V$ is the *data marking*. The initial marking of a Petri net with Data DPN is $(M_{init}, init)$. Recall that *init* is a function that defines the initial value for each variable.

$(DPN, (M, s))[b]$ denotes that a binding b is enabled in marking (M, s) , which indicates that each of its input places $\bullet t$ contains at least one token (control-flow enabled), b is valid and and $s|_{read(t)} = r$ (the actual values read match the binding).⁴

³ $\mathcal{P}(X)$ is the powerset of X , i.e., $Y \in \mathcal{P}(X)$ is and only if $Y \subseteq X$.

⁴ $f|_Q$ is the function projected on Q : $dom(f|_Q) = dom(f) \cap Q$ and $f|_Q(x) = f(x)$ for $x \in dom(f|_Q)$. Projection can also be used for bags and sequences, e.g., $[x^3, y, z^2]|_{\{x,y\}} = [x^3, y]$ and $\langle y, z, y \rangle|_{\{x,y\}} = \langle y, y \rangle$.

Table 1. Definitions of the guards of the transitions in Fig. 3. Variables and transition names are abbreviated as described in Figure 1. Subscripts r and w refer to, respectively, the values read and written for that given variable.

Transition	Guard
Credit Request	<code>true</code>
Verify	$0.1 \cdot A_r < I_w < 0.2 \cdot A_r$
Assessment	$V_R = \text{true}$
Register Negative Verification	$V_r = \text{false} \wedge D_w = \text{false}$
Inform Requester	<code>true</code>
Renegotiate Request	$V_r = \text{false} \wedge A_w < A_r$
Register Negative Request	$D_r = \text{false}$
Open Credit	$D_r = \text{true}$

An enabled binding $b = (t, r, w)$ may *occur*, i.e., one token is removed from each of the input places $\bullet t$ and one token is produced for each of the output places $t\bullet$. Moreover, the variables are updated as specified by w . Formally: $M' = (M \setminus \bullet t) \uplus t\bullet$ is the control-flow marking resulting from firing enabled transition t in marking M (abstracting from data) and $s' = s \oplus w$ is the data marking where $s'(v) = w(v)$ for all $v \in \text{write}(t)$ and $s'(v) = s(v)$ for all $v \in V \setminus \text{write}(t)$. $(DPN, (M, s))[b](DPN, (M', s'))$ denotes that b is enabled in (M, s) and the occurrence of b results in marking (M', s') .

Figure 3 shows a Petri net with Data DPN_{ex} that models the same process as represented in Figure 1 as BPMN diagram, and Table 1 illustrates the conditions of the guards of the transitions of DPN_{ex} . The labeling function l is such that the domain of l is the set of transitions of DPN_{ex} and, for each transition t of DPN_{ex} , $l(t) = t$. In other words, the set of activity labels coincides with the set of transitions.

Let $\sigma_b = \langle b_1, b_2, \dots, b_n \rangle$ be a sequence of bindings. $(DPN, (M, s))[\sigma_b](DPN, (M', s'))$ denotes that there is a set of markings $(M_0, s_0), (M_1, s_1), \dots, (M_n, s_n)$ such that $(M_0, s_0) = (M, s)$, $(M_n, s_n) = (M', s')$, and $(DPN, (M_i, s_i))[b_{i+1}](DPN, (M_{i+1}, s_{i+1}))$ for $0 \leq i < n$. A marking (M', s') is *reachable* from (M, s) if there exists a σ_b such that $(DPN, (M, s))[\sigma_b](DPN, (M', s'))$.

$\phi_f(DPN) = \{\sigma_b \mid \exists_s (DPN, (M_{init}, init))[\sigma_b](DPN, (M_{final}, s))\}$ is the *set of complete binding sequences*, thus describing the behavior of DPN .

Definition 7 (Union of Petri nets with Data). Let $DPN^1 = (SN^1, V^1, val^1, init^1, read^1, write^1, guard^1)$ and $DPN^2 = (SN^2, V^2, val^2, init^2, read^2, write^2, guard^2)$ with $V^1 \cap V^2 = \emptyset$. $DPN^1 \cup DPN^2 = (SN^1 \cup SN^2, V^1 \cup V^2, val^1 \oplus val^2, init^1 \oplus init^2, read^3, write^3, guard^3)$ is the union such that

- $read^3(t) = read^1(t)$, $write^3(t) = write^1(t)$, and $guard^3(t) = guard^1(t)$ if $t \in T^1 \setminus T^2$,
- $read^3(t) = read^2(t)$, $write^3(t) = write^2(t)$, and $guard^3(t) = guard^2(t)$ if $t \in T^2 \setminus T^1$, and
- $read^3(t) = read^1(t) \cup read^2(t)$, $write^3(t) = write^1(t) \cup write^2(t)$, and $guard^3(t) = guard^1(t) \cdot guard^2(t)$ if $t \in T^1 \cap T^2$.

2.3 Event Logs and Relating Models to Event Logs

Next we introduce *event logs* and relate them to the *observable* behavior of a *DPN*.

Definition 8 (Trace, Event Log with Data). A trace $\sigma \in (\mathcal{U}_A \times \mathcal{U}_{VM} \times \mathcal{U}_{VM})^*$ is a sequence of activities with input and output data. $L \in \mathbb{B}((\mathcal{U}_A \times \mathcal{U}_{VM} \times \mathcal{U}_{VM})^*)$ is an event log with read and write information, i.e., a multiset of traces with data.

Definition 9 (From Bindings to Traces). Consider a Petri net with Data with transitions T and labeling function $l \in T \not\rightarrow \mathcal{U}_A$. A binding sequence $\sigma_b \in (T \times \mathcal{U}_{VM} \times \mathcal{U}_{VM})^*$ can be converted into a trace $\sigma_v \in (\mathcal{U}_A \times \mathcal{U}_{VM} \times \mathcal{U}_{VM})^*$ by removing the bindings that correspond to unlabeled transitions and by mapping the labeled transitions onto their corresponding label. $l(\sigma_b)$ denotes the corresponding trace σ_v .

Note that we overload the labeling function to binding sequences, $\sigma_v = l(\sigma_b)$. This is used to define $\phi(DPN)$: the set of all visible traces.

Definition 10 (Observable Behavior of a Petri net with Data). Let *DPN* be a Petri net with Data. $(DPN, (M, s))[\sigma_v \triangleright (DPN, (M', s'))]$ if and only if there is a sequence σ_b such that $(DPN, (M, s))[\sigma_b](DPN, (M', s'))$ and $\sigma_v = l(\sigma_b)$. $\phi(DPN) = \{l(\sigma_b) \mid \sigma_b \in \phi_f(DPN)\}$ is the set of visible traces starting in $(M_{init}, init)$ and ending in (M_{final}, s) for some data marking s .

Definition 11 (Perfectly Fitting with Data). A trace $\sigma \in (\mathcal{U}_A \times \mathcal{U}_{VM} \times \mathcal{U}_{VM})^*$ is perfectly fitting $DPN \in \mathcal{U}_{DPN}$ if $\sigma \in \phi(DPN)$. An event log $L \in \mathbb{B}((\mathcal{U}_A \times \mathcal{U}_{VM} \times \mathcal{U}_{VM})^*)$ is perfectly fitting *DPN* if all of its traces are perfectly fitting.

Later, we will need to project binding sequences and traces onto subsets of transitions/activities and variables. Therefore, we introduce a generic projection operator $\Pi_{Y,V}(\sigma)$ that removes transitions/activities not in Y and variables not in V .

Definition 12 (Projection). Let X be a set of transitions or activities (i.e., $X \subseteq T$ or $X \subseteq \mathcal{U}_A$). Let $Y \subseteq X$ be a subset and $V \subseteq \mathcal{U}_{VN}$ a subset of variable names. Let $\sigma \in (X \times \mathcal{U}_{VM} \times \mathcal{U}_{VM})^*$ be a binding sequence or a trace with data. $\Pi_{Y,V}(\sigma) \in (Y \times (V \not\rightarrow \mathcal{U}_{VV}) \times (V \not\rightarrow \mathcal{U}_{VV}))^*$ is the projection of σ onto transitions/activities Y and variables V . Bindings/events unrelated to transitions/activities in Y are removed completely. Moreover, for the remaining bindings/events all read and write variables not in V are removed. $\Pi_{Y,V}(L) = [\Pi_{Y,V}(\sigma) \mid \sigma \in L]$ lifts the projection operator to the level of logs.

2.4 Alignments

Conformance checking requires an *alignment* of event log L and process model *DPN*, that is the alignment of each single trace $\sigma \in L$ and process model *DPN*.

The events in the event log need to be related to transitions in the model, and vice versa. Such an alignment shows how the event log can be replayed on the process model. Building this alignment is far from trivial, since the log may deviate from the model at an arbitrary number of places. We need to relate “moves” in the log to “moves” in the model in order to establish an alignment between a process model and an event log. It may be that some of the moves in the log cannot be mimicked by the model and vice versa. We denote such “no moves” by \gg . An alignment is a sequence of moves:

Table 2. Examples of complete alignments of $\sigma_{example}$ and N . For readability, the read operations are omitted. Of course, read operations for any variable must match the most recent value for that variable. Any move is highlighted with a gray color if it contains deviations, i.e. it is not a move in both without incorrect read/write operations.

(a)	(b)		
Event-Log Trace	Process	Event-Log Trace	Process
(a, {(A,4000)})	(a, {(A,4000)})	(a, {(A,5100)})	(a, {(A,5100)})
(b, {(I,450),(V,false)})	(b, {(I,450),(V,true)})	(b, {(I,511),(V,true)})	(b, {(I,511),(V,true)})
(c, {(D,true)})	(c, {(D,true)})	(c, {(D,true)})	(c, {(D,true)})
(e, \emptyset)	(e, \emptyset)	(e, \emptyset)	(e, \emptyset)
(f, {(A,5000)})	(f, {(A,3000)})	(f, {(A,5000)})	(f, {(A,5000)})
(b, {(I,450),(V,false)})	(b, {(I,450),(V,false)})	(b, {(I,511),(V,false)})	(b, {(I,511),(V,false)})
(d, {(D,false)})	(d, {(D,false)})	(d, {(D,false)})	(d, {(D,false)})
(e, \emptyset)	(e, \emptyset)	(e, \emptyset)	(e, \emptyset)
(h, \emptyset)	\gg	(h, \emptyset)	\gg
	(g, \emptyset)		(g, \emptyset)

Definition 13 (Legal alignment moves). Let $DPN = (SN, V, val, init, read, write, guard)$ be a Petri net with Data, with $SN = (PN, M_{init}, M_{final})$ and $PN = (P, T, F, l)$. Let $S_L = \mathcal{U}_A \times \mathcal{U}_{VM} \times \mathcal{U}_{VM}$ be the universe of events. Let $S_{DPN} = T \times \mathcal{U}_{VM} \times \mathcal{U}_{VM}$ be the universe of bindings of DPN. Let be $S_{DPN}^{\gg} = S_{DPN} \cup \{\gg\}$ and $S_L^{\gg} = S_L \cup \{\gg\}$.

A legal move in an alignment is represented by a pair $(s_L, s_M) \in (S_L^{\gg} \times S_{DPN}^{\gg}) \setminus \{(\gg, \gg)\}$ such that

- (s_L, s_M) is a move in log if $s_L \in S_L$ and $s_M = \gg$,
- (s_L, s_M) is a move in model if $s_L = \gg$ and $s_M \in S_{DPN}$,
- (s_L, s_M) is a move in both without incorrect read/write operations if $s_M = (t, r, w) \in S_{DPN}$ and $s_L = (l(t), r, w) \in S_L$,
- (s_L, s_M) is a move in both with incorrect read/write operations if $s_M = (t, r, w) \in S_{DPN}$ and $s_L = (l(t), r', w') \in S_L$, and $r \neq r'$ or $w \neq w'$.

All other moves are considered as illegal.

Definition 14 (Alignments). Let $DPN = (SN, V, val, init, read, write, guard)$ be a Petri net with Data and $\sigma \in (S_L)^*$ be an event-log trace. Let \mathcal{A}_{DPN} be the set of legal moves for DPN. A complete alignment of σ_L and DPN is a sequence $\gamma \in \mathcal{A}_{DPN}^*$ such that, ignoring all occurrences of \gg , the projection on the first element yields σ_L and the projection on the second yields a $\sigma_P \in \phi_f(DPN)$.

Table 2 shows two complete alignments of the process model in Figure 3 and the log trace σ_{ex} from Section 1.

In order to define the severity of a deviation, we introduce a cost function on legal moves: $\kappa \in \mathcal{A}_{DPN} \rightarrow \mathbb{R}_0^+$. This cost function can be used to favor one type of explanation for deviations over others. The cost of each legal move depends on the specific model and process domain and, hence, the cost function κ needs to be defined specifically for each setting. The cost of an alignment γ is the sum of the cost of all individual moves composing it: $\mathcal{K}(\gamma) = \sum_{(s_L, s_M) \in \gamma} \kappa(s_L, s_M)$.

However, we do not aim to find just any complete alignment. Our goal is to find a complete alignment of σ_L and DPN which minimizes the cost: an optimal alignment. Let $\Gamma_{\sigma_L, N}$ be the (infinite)set of all complete alignments of σ_L and DPN. The alignment $\gamma \in \Gamma_{\sigma_L, DPN}$ is an *optimal alignment* if, for all $\gamma' \in \Gamma_{\sigma_L, N}$, $\mathcal{K}(\gamma) \leq \mathcal{K}(\gamma')$. Note

that an optimal alignment does not need to be unique, i.e. multiple complete alignments with the same minimal cost may exist.

Let us consider again our example introduced above. Let us assume to have a cost function κ^s such that $\kappa^s(s_L, s_M) = 1$ if (s_L, s_M) is a visible move in process or a move in log (i.e. $s_L \Rightarrow\!\!>$ and s_M corresponds to a labeled transition or, conversely, $s_M \Rightarrow\!\!>$, respectively) or a move in both with incorrect read/write operations and $\kappa^s(s_L, s_M) = 0$ in case of move in both without incorrect read/write operations or a move in model corresponding to an unlabeled transition. The alignment in Table 2a has a cost of 6 whereas the alignment in Table 2b has a cost 8.⁵ It follows that the former is a better alignment. As a matter of fact, it is also an optimal alignment, although it is not the only one. For instance, any variation of such an alignment where the move for f is of the form (now including read operations) $((\mathbf{f}, \{(A, 4000)\}, \{(A, 5000)\}) (\mathbf{f}, \{(A, 4000)\}, \{(A, x)\}))$ with $2250 < x < 4000$ corresponds to an optimal alignment, as well.

In Section 1, we have mentioned that the data-aware conformance checking is undecidable in the general case. This is caused by the fact that Petri nets with Data are Turing-complete. Therefore, it is not decidable to verify whether a sequence of valid bindings exists that takes from the initial marking to any final marking (M_{final}, s) . As a consequence, for instance, it is not possible to find an alignment of a Petri net with Data and the empty log trace. As mentioned in Section 1, the problem becomes decidable (with an exponential complexity) if guards are restricted to linear (in)equalities.

3 Valid Decomposition of Data-Aware Models

In [6] the author defines *valid decomposition* in terms of Petri nets: the overall system net SN is decomposed into a collection of subnets $\{SN^1, SN^2, \dots, SN^n\}$ such that the union of these subnets yields the original system net. A decomposition is *valid* if the subnets “agree” on the original labeling function (i.e., the same transition always has the same label), each place resides in just one subnet, and also each invisible transition resides in just one subnet. Moreover, if there are multiple transitions with the same label, they should reside in the same subnet. Only unique visible transitions can be shared among different subnets.

Definition 15 (Valid Decomposition for Petri nets [6]). *Let $SN \in \mathcal{U}_{SN}$ be a system net with labeling function l . $D = \{SN^1, SN^2, \dots, SN^n\} \subseteq \mathcal{U}_{SN}$ is a valid decomposition if and only if:*

- $SN^i = (N^i, M_{init}^i, M_{final}^i)$ is a system net with $N^i = (P^i, T^i, F^i, l^i)$ for all $1 \leq i \leq n$,
- $l^i = l|_{T^i}$ for all $1 \leq i \leq n$,
- $P^i \cap P^j = \emptyset$ for $1 \leq i < j \leq n$,
- $T^i \cap T^j \subseteq T_v^u(SN)$ for $1 \leq i < j \leq n$,
- $rng(l^i) \cap rng(l^j) \subseteq T_v^u(SN)$ for $1 \leq i < j \leq n$, and
- $SN = \bigcup_{1 \leq i \leq n} SN^i$.

$\mathcal{D}(SN)$ is the set of all valid decompositions of SN .

⁵ They also include a cost of two that is accounted for incorrect read operations, not shown in the alignments, which are caused by incorrect write operations.

From the definition the following properties follow:

1. each place appears in precisely one of the subnets, i.e., for any $p \in P$: $|\{1 \leq i \leq n \mid p \in P^i\}| = 1$,
2. each invisible transition appears in precisely one of the subnets, i.e., for any $t \in T \setminus T_v(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| = 1$,
3. all visible transitions with the same label (i.e. the label is not unique) appear in the same subnet, i.e., for any $a \in A_v(SN) \setminus A_v^u(SN)$: $|\{1 \leq i \leq n \mid \exists t \in T_v(SN) \cap T^i, l(t)\}| = 1$,
4. visible transitions having a unique label may appear in multiple subnets, i.e., for any $t \in T_v^u(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| \geq 1$, and
5. each edge appears in precisely one of the subnets, i.e., for any $(x, y) \in F$: $|\{1 \leq i \leq n \mid (x, y) \in F^i\}| = 1$.

As shown in [6], these observations imply that conformance checking can be decomposed. Any trace that fits the overall process model can be decomposed into smaller traces that fit the individual model fragments. Moreover, if the smaller traces fit the individual fragments, then they can be composed into a trace that fits into the overall process model. This result is the basis for decomposing process mining problems.

Theorem 1 (Conformance Checking Can be Decomposed [6]). *Let $L \in \mathbb{B}(A^*)$ be an event log with $A \subseteq \mathcal{U}_A$ and let $SN \in \mathcal{U}_{SN}$ be a system net. For any valid decomposition $D = \{SN^1, SN^2, \dots, SN^n\} \subseteq \mathcal{D}(SN)$: L is perfectly fitting system net SN if and only if for all $1 \leq i \leq n$: the projection of L onto $A_v(SN^i)$ is perfectly fitting SN^i .*

In this paper, the definition of valid decomposition is extended to cover Petri nets with data.

Definition 16 (Valid Decomposition for Petri nets with Data). *Let $DPN \in \mathcal{U}_{DPN}$ be a Petri net with Data. $D = \{DPN^1, DPN^2, \dots, DPN^n\} \subseteq \mathcal{U}_{DPN}$ is a valid decomposition if and only if:*

- for all $1 \leq i \leq n$: $DPN^i = (SN^i, V^i, val^i, init^i, read^i, write^i, guard^i)$ is a Petri net with Data, $SN^i = (PN^i, M_{init}^i, M_{final}^i) \in \mathcal{U}_{SN}$ is a system net, and $PN^i = (P^i, T^i, F^i, l^i)$ is a labeled Petri net,
- $D' = \{SN^1, SN^2, \dots, SN^n\} \subseteq \mathcal{U}_{SN}$ is a valid decomposition of $\bigcup_{1 \leq i \leq n} SN^i$,
- $V^i \cap V^j = \emptyset$ for $1 \leq i < j \leq n$,
- $DPN = \bigcup_{1 \leq i \leq n} DPN^i$.

$\mathcal{D}(DPN)$ is the set of all valid decompositions of DPN .

Each variable appears in precisely one of the subnets. Therefore, there cannot be two fragments that read and or write the same data variables: $\bigcup_{t \in T^i} read^i(t) \cup write^i(t) \cap \bigcup_{t \in T^j} read^j(t) \cup write^j(t) = \emptyset$ for $1 \leq i < j \leq n$. Moreover, two guards in different fragments cannot refer to the same variable. If a transition t appears in multiple fragments, then it needs to have a visible unique label as shown in [6]. Such a uniquely labeled transition t shared among fragments, may use, read, or write different variables in different fragments. Since $DPN = \bigcup_{1 \leq i \leq n} DPN^i$, we know that, for all t in DPN , $guard(t)$ is the product of all $guard^i(t)$ such that $t \in T^i$. Without loss of generality we can assume that the first k fragments share t . Hence, $guard(t) = guard^1(t) \cdot \dots \cdot guard^k(t)$. Hence, in a valid decomposition, the guard of a shared transition can only

be split if the different parts do not depend on one another. Notice that, the splitting of the data variables is limited by how the variables are used throughout the process, existing a worst-case where all the data variables are used in all the steps of the process.

Based on these observations, we prove that we can decompose conformance checking also for Petri nets with data.

Theorem 2 (Conformance Checking With Data Can be Decomposed). *Let $L \in \mathcal{B}((\mathcal{U}_A \times \mathcal{U}_{VM} \times \mathcal{U}_{VM})^*)$ be an event log with information about reads and writes and let $DPN \in \mathcal{U}_{DPN}$ be a Petri net with Data. For any valid decomposition $D = \{DPN^1, DPN^2, \dots, DPN^n\} \subseteq \mathcal{U}_{DPN}$: L is perfectly fitting Petri net with Data DPN if and only if for all $1 \leq i \leq n$: $\Pi_{A_v(SN^i), V^i}(L)$ is perfectly fitting DPN^i .*

Proof. Let $DPN = (SN, V, val, init, read, write, guard)$ be a Petri net with Data with $SN = (PN, M_{init}, M_{final})$ and $PN = (P, T, F, l)$. Let $D = \{DPN^1, DPN^2, \dots, DPN^n\}$ be a valid decomposition of DPN with $DPN^i = (SN^i, V^i, val^i, init^i, read^i, write^i, guard^i)$, $SN^i = (PN^i, M_{init}^i, M_{final}^i) \in \mathcal{U}_{SN}$, and $PN^i = (P^i, T^i, F^i, l^i)$.

(\Rightarrow) Let $\sigma_v \in L$ be such that there exists a data marking s such that $(DPN, (M_{init}, init))[\sigma_v \triangleright (DPN, (M_{final}, s))]$. This implies that there exists a corresponding σ_b with $(DPN, (M_{init}, init))[\sigma_b \rangle (DPN, (M_{final}, s))$ and $l(\sigma_b) = \sigma_v$. For all $1 \leq i \leq n$, we need to prove that there is a σ_b^i with $(DPN^i, (M_{init}^i, init^i))[\sigma_b^i \rangle (DPN^i, (M_{final}^i, s^i))$ for some s^i . This follows trivially because DPN^i can mimic any move of DPN with respect to transitions T^i : just take $\sigma_b^i = \Pi_{T^i, V^i}(\sigma_b)$. Note that guards can only become weaker by projection.

(\Leftarrow) Let $\sigma_v \in L$. For all $1 \leq i \leq n$, let σ_b^i be such that $(DPN^i, (M_{init}^i, init^i))[\sigma_b^i \rangle (DPN^i, (M_{final}^i, s^i))$ and $l^i(\sigma_b^i) = \Pi_{A_v(SN^i), V^i}(\sigma_v)$. The different σ_b^i sequences can be stitched together into an overall σ_b s.t. $(DPN, (M_{init}, init))[\sigma_b \rangle (DPN, (M_{final}, s))$ with $s = s^1 \oplus s^2 \oplus \dots \oplus s^n$. This is possible because transitions in one subnet can only influence other subnets through unique visible transitions and these can only move synchronously as defined by σ_v . Moreover, guards can only be split in independent parts (see Definition 16). Suppose that t appears in T_i and T_j , then $guard(t) = guard^i(t) \cdot guard^j(t)$. Hence, a read/write in subnet i cannot limit a read/write in subnet j . Therefore, we can construct σ_b and $l(\sigma_b) = \sigma_v$. \square

4 SESE-Based Strategy for Realizing a Valid Decomposition

In this section we present a concrete strategy to instantiate the valid decomposition definition over a Petri net with data presented in the previous section (cf. Def.16). The proposed strategy decomposes the Petri net with data in a number of Single-Entry Single-Exit (SESE) components, which have recently been shown to create meaningful fragments of a process model [11,8]. SESE decomposition is indicated for well-structured models, whereas for unstructured models some automatic transformation techniques can be considered as a pre-processing step [12].

We will now informally describe the necessary notions for understanding the proposed data-oriented SESE-based valid decomposition strategies described below. For the sake of clarity, we will focus on the control flow to illustrate the concepts, although the definitions will be extended at the end to also consider data.

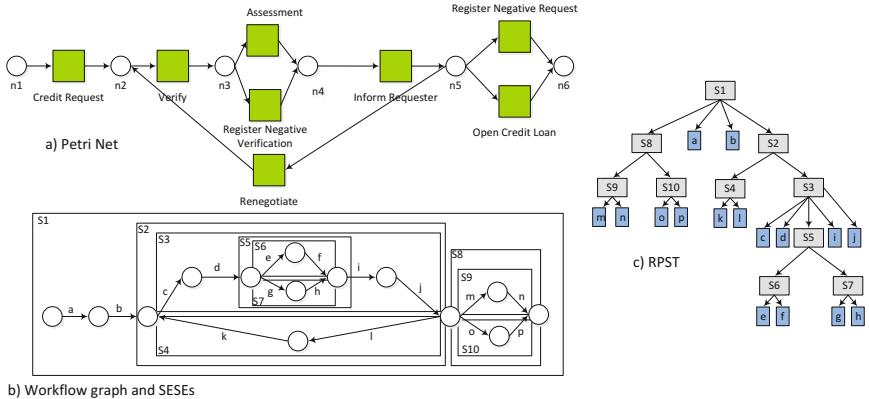


Fig. 4. A Petri net modeling the control-flow of the running example, its workflow graph and the RPST and SESE decomposition

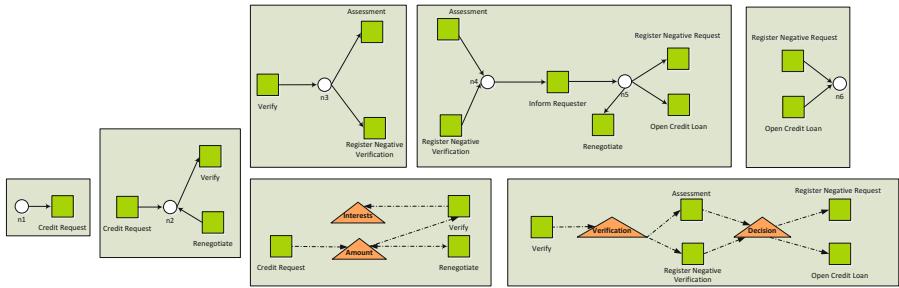


Fig. 5. SESE-based decomposition for the running example, with 2-decomposition

Given Petri net $PN = (P, T, F, l)$, its *workflow graph* is the structural graph $WG = (S, E)$ with no distinctions between places and transitions, i.e., $S = P \cup T$ and $E = F$. For instance, Fig. 4(b) shows the workflow graph of the Petri net of Fig. 4(a) (corresponding with the control-flow part of the running example). Given a subset of edges $E' \subseteq E$ of WG , the nodes $S|_{E'} = \{s \in S : \exists s' \in S. (s, s') \in E' \vee (s', s) \in E'\}$ can be partitioned into interior and boundary. Interior nodes have no connection with nodes outside $S|_{E'}$, while boundary nodes do. Furthermore, boundary nodes can be partitioned into entry (no incoming edge belongs to E), or exit (no outgoing edge belongs to E). $E' \subseteq E$ is a *SESE* of WG iff the subnet derived from E has exactly two boundary nodes: one entry and one exit. Fig. 4(b) shows all non-trivial SESEs⁶ of the Petri net of Fig. 4(a). For a formal definition we refer to [11].

The decomposition based on SESEs is a well studied problem in the literature, and can be computed in linear time. In [13,14], efficient algorithms for constructing the *Refined Process Structure Tree (RPST)*, i.e., a hierarchical structure containing all the canonical SESEs of a model, were presented. Informally, an RPST is a tree where the nodes are canonical SESEs, such that the parent of a SESE S is the smallest SESE that

⁶ Note that by definition, a single edge is a SESE.

Algorithm 1. SESE-based Decomposition

-
- 1: Build data workflow graph DWG from F, R, W
 - 2: Compute $RPST$ from DWG
 - 3: Compute SESE decomposition D from the $RPST$
 - 4: Compute and merge subnets if necessary to preserve valid decomposition.
 - 5: **return** valid decomposition where perspectives are decomposed altogether
-

contains S . Fig. 4(c) shows the RPST of the workflow graph depicted in Fig. 4(b). By selecting a particular set of SESEs in the RPST (e.g., k-decomposition [8]), it is possible to obtain a partitioning of the arcs. We refer the reader to the aforementioned work for a formal description of the SESE-based decomposition.

To extend the previous definitions to also account for data, one simply has to incorporate in the workflow graph the variables and read/write arcs, i.e., the *data workflow graph* of a Petri net with Data $((P, T, F, l), M_{init}, M_{final}), V, val, init, read, write, guard$ with data arcs $R = \{(v, t) | v \in read(t)\}$ and $W = \{(t, v) | v \in write(t)\}$ is $DWG = (S, E)$ with $S = P \cup T \cup V$ and $E = F \cup R \cup W$. The subsequent definitions after this extension (SESE, RPST) are analogous.

Similar to [8], we propose a SESE decomposition to analyze the conformance of Petri nets with data, but considering data workflow graph instead. Algorithm 1 describes the steps necessary to construct a SESE decomposition. The arcs are partitioned in SESEs by means of creating the RPST from the data workflow graph, and selecting a particular set of SESES over it. Once the partitioning is done, a subnet is created for each part. Subnets contradicting some of the requirements of Def. 16 (e.g. sharing places, invisible or duplicate transitions, variables, or transitions with non-splitting guards) are merged to preserve the valid decomposition definition.

Figure 5 shows the decomposition for the example of Fig.3, where the RPST is partitioned using the 2-decomposition algorithm [8], i.e., SESEs of at most 2 arcs⁷. To ensure a valid decomposition is obtained, step 4 of Algorithm 1 combines multiple SESE fragments into larger fragments, which are not necessarily SESEs anymore.

5 Implementation and Experimental Results

The approach discussed in this paper has been implemented as a plug-in for the open-source *ProM* framework for process mining.⁸ Our plug-in requires a Petri Net with Data and an event log as input and returns as many bags of alignments as the number of fragments in which the Petri Net with Data has been decomposed. Each bag refers to a different fragment and shows the alignments of each log trace and that fragment. A second type of output is also produced in which the alignments' information is projected onto the Petri net with Data. Transitions are colored according to the number of deviations: if no deviation occurs for a given transition, the respective box in the model is white-colored. The filling color of a box shades towards red as a larger fraction of deviations occur for the corresponding transition. Something similar is also done for

⁷ Although the SESEs have at most two arcs, this is not guaranteed for the final subnets, i.e., some subnets are merged to preserve the valid decomposition definition.

⁸ <http://www.promtools.org>

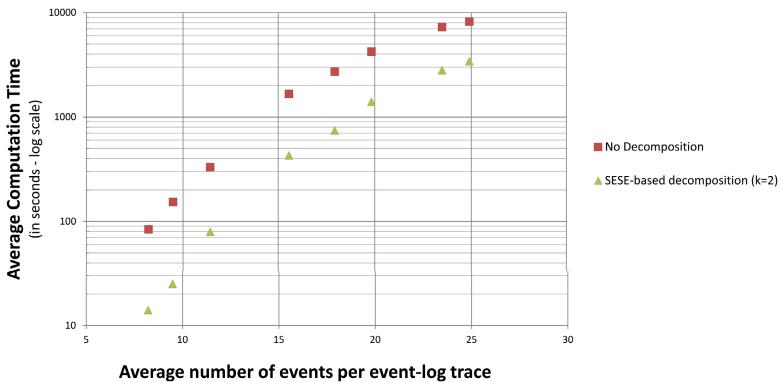


Fig. 6. Computation time for checking the conformance of the Petri net with Data in Figure 3 and event logs of different size. The Y axis is on a logarithmic scale.

variables: the more incorrect read/write operations occur for a variable, the more the variable is shown with a color close to red. This output is extremely interesting from an end-user viewpoint as it allows for gaining a helicopter view on the main causes of deviations. Space limitations prevent us from giving more details and showing screenshots of the output. Interested readers can refer to [15] for further information.

As previously mentioned, the plug-in has been evaluated using a number of synthetic event logs and also a real-life process. The plug-in has been evaluated using the model in Figure 3 and with a number of event logs that were artificially generated. In particular, we have generated different event logs with the same number of traces, 5000, but increasing number of events, meaning that, on average, traces were of different length. To simulate that, for each simulated process execution, an increasing number of renegotiations was enforced to happen. Traces were also generated so as to contain a number of deviations: the event logs were generated in a way that 25% of transitions fired violating the guards.

Figure 6 shows the results of checking for conformance of the different event logs and the process model, comparing the SESE-based decomposition with $k = 2$ with the case in which no decomposition is made. To check the conformance of each fragment, we used the technique reported in [4]. Each dot in the chart indicates a different event log with traces of different size. The computation time refers to the conformance checking of the whole event logs (i.e., 5000 traces). The decomposed net is the same as in Figure 5. Regarding the cost function, we assign cost 1 to any deviation; however, this could be customized based on domain knowledge. The results show that, for every combination of event log and process model, the decomposition significantly reduces the computation time and the improvement is exponential in the size of the event log.

To assess the practical relevant of the approach, we also performed an evaluation with a Dutch financial institute. The process model was provided by a process analyst of the institute and consists of 21 transitions: 13 transitions with unique labels, 3 activities labels shared between 2 transitions (i.e. 6 transitions in total), plus 3 invisible transitions. The model contains twelve process variables, which are read and written by the activities when being executed. The process model is omitted for space reasons

and shown in [15]. We were also provided with an event log that recorded the execution of 111 real instances of such a process; overall, the 111 log traces contained 3285 events, which means roughly 29.6 events per trace. We checked the conformance of this process model and this event log, comparing the results when the model has or has not been decomposed in small fragments. For conformance checking, here we used the technique reported in [5] since the provided process model breaks the soundness assumptions required by [4]. For this experiment round, the additional optimizations proposed in [5] were deactivated to allow for a fair comparison.

The application of the decomposition approach to this real-life case study has shown tremendous results: the conformance checking has required 52.94 seconds when the process model was decomposed using the SESE-based technique presented in Section 4; conversely, it required 52891 seconds when the model was not decomposed. This indicates that decomposing the process model allowed us to save 99.999% of the computation time. As a matter of fact, we tried for different values of SESE parameter k but we obtained similar results: the computation time did not move away for more than 1 second. The reason of this is related to the fact that every decomposition for any value of k always contained a certain fragment, along with others. Indeed, that fragment could not be decomposed any further than a given extent. Since the computation time was mostly due to constructing alignments with that fragment, no significant difference in computation time could be observed when varying k .

6 Conclusions and Future Work

Conformance checking is becoming more important for two reasons: (1) the volume of event data available for checking normative models is rapidly growing (the topic of “Big Data” is on the radar of all larger organizations) and (2) because of a variety of regulations there is a need to check compliance. Moreover, conformance checking is also used for the evaluation of process discovery algorithms. Also genetic process mining algorithms heavily rely on the efficiency of conformance checking techniques.

Thus far, lion’s share of conformance checking techniques has focused on control-flow and relatively small event logs. As shown in this paper, abstracting from other perspectives may lead to misleading conformance results that are too optimistic. Moreover, as process models and event logs grow in size, divide-and-conquer approaches are needed to still be able to check conformance and diagnose problems. Perspectives such as work distribution, resource allocation, quality of service, temporal constraints, etc. can all be encoded as data constraints. Hence, there is an urgent need to support data-aware conformance checking in-the-large.

This paper demonstrates that data-aware decompositions can be used to speed up conformance checking significantly. The evaluation with a real-life case study has shown that real data-aware process models can indeed be decomposed, thus obtaining even tremendous saving of computation time. As future work, we would like to extend our experimental evaluation with real-life process models of larger sizes. Moreover, we would like to explore alternative decomposition strategies using properties of the underlying data, and to analyze the impact of different component sizes. This paper only focuses on fitness aspect of conformance, namely whether a trace can be replayed on a process model. However, recently, research has also been carried on as regards to different conformance dimensions [16,17], such as whether the model is precise enough

to not allow for too much behavior compared with what observed in reality in the event log. We plan to use data-aware decomposition approaches to speed up the assessment of the quality of process models with respect to these other conformance dimensions, as well.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. *Information System* 33(1), 64–95 (2008)
3. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2011), pp. 55–64. IEEE Computer Society (2011)
4. de Leoni, M., van der Aalst, W.M.P.: Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 113–129. Springer, Heidelberg (2013)
5. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced Multi-Perspective Checking of Process Conformance. BPM Center Report BPM-14-07 (2014)
6. van der Aalst, W.M.P.: Decomposing Petri nets for process mining: A generic approach. *Distributed and Parallel Databases* 31(4), 471–507 (2013)
7. van der Aalst, W.M.P.: Decomposing process mining problems using passages. In: Haddad, S., Pomello, L. (eds.) PETRI NETS 2012. LNCS, vol. 7347, pp. 72–91. Springer, Heidelberg (2012)
8. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Single-entry single-exit decomposed conformance checking. *Information Systems* 46, 102–122 (2014)
9. Montali, M., Chesani, F., Mello, P., Maggi, F.M.: Towards data-aware constraints in declare. In: Shin, S.Y., Maldonado, J.C. (eds.) SAC, pp. 1391–1396. ACM (2013)
10. Jensen, K., Kristensen, L.: Coloured Petri Nets. Springer (2009)
11. Polyvyanyy, A.: Structuring process models. PhD thesis, University of Potsdam (2012)
12. Dumas, M., García-Bañuelos, L., Polyvyanyy, A.: Unraveling unstructured process models. In: Mendling, J., Weidlich, M., Weske, M. (eds.) BPMN 2010. LNBP, vol. 67, pp. 1–7. Springer, Heidelberg (2010)
13. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. *Data Knowl. Eng.* 68(9), 793–818 (2009)
14. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: Bravetti, M. (ed.) WS-FM 2010. LNCS, vol. 6551, pp. 25–41. Springer, Heidelberg (2011)
15. de Leoni, M., Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Decomposing Conformance Checking on Petri Nets with Data. BPM Center Report BPM-14-06 (2014)
16. Munoz-Gama, J., Carmona, J.: A General Framework for Precision Checking. *International Journal of Innovative Computing, Information and Control (IJICIC)* 8(7B), 5317–5339 (2012)
17. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information System* 37(7), 654–676 (2012)

A Pattern Approach to Conquer the Data Complexity in Simulation Workflow Design

Peter Reimann, Holger Schwarz, and Bernhard Mitschang

Institute of Parallel and Distributed Systems, University of Stuttgart, Germany

Abstract. Scientific workflows may be used to enable the collaborative implementation of scientific applications across various domains. Since each domain has its own requirements and solutions for data handling, such workflows often have to deal with a highly heterogeneous data environment. This results in an increased complexity of workflow design. As scientists typically design their scientific workflows on their own, this complexity hinders them to concentrate on their core issue, namely the experiments, analyses, or simulations they conduct. In this paper, we present a novel approach to a pattern-based abstraction support for the complex data management in simulation workflows that goes beyond related work in similar research areas. A pattern hierarchy with different abstraction levels enables a separation of concerns according to the skills of different persons involved in workflow design. The goal is that scientists are no longer obliged to specify low-level details of data management in their workflows. We discuss the advantages of this approach and show to what extent it reduces the complexity of simulation workflow design. Furthermore, we illustrate how to map patterns onto executable workflows. Based on a prototypical implementation of three real-world simulations, we evaluate our approach according to relevant requirements.

1 Introduction

Nowadays, *scientific workflows* are increasingly adopted to enable the collaborative implementation of scientific applications across various domains [24]. This includes applications such as experiments, data analyses, or computer-based simulations. *Simulation workflows*, as a sub-area of scientific workflows, are typically compositions of long-running numeric calculations [5]. These calculations realize mathematical simulation models, e. g., based on partial differential equations. To make simulations more realistic and their outcome more precise, scientists couple simulation models from several scientific domains in simulation workflows [8]. This allows them to cover various levels of granularity in simulation calculations. The calculations for each of the simulation models have to process huge data sets from diverse data sources and in multiple proprietary data formats [15]. Furthermore, the results from one simulation model are often used as input for other simulation models. As different simulation models and underlying simulation tools typically rely on different solutions for data handling, this even multiplies the complexity of data management in simulation workflows. So, these workflows have to embed many sophisticated data provisioning and data integration tasks.

As an example, consider a simulation of structure changes in human bones, which combines a bio-mechanical simulation on the macro scale with a systems-biology simulation on the micro scale [8]. The workflow realizing this coupled simulation consists of more than 20 activities [15]. Two thirds of these activities deal with service calls and the complex data management necessary to provide data for these services. Today, simulation workflow design is typically carried out by the scientists themselves. Hence, they have to specify many low-level details of data provisioning, but they usually do not have the necessary skills in workflow design or data management. This hinders scientists to concentrate on their core issues, namely the development of mathematical simulation models, the execution of simulation calculations, and the interpretation of their results.

In a previous publication, we have presented the vision of a pattern-based approach to make simulation workflow design tailor-made for scientists [15]. This approach goes beyond related work for artifact-centric business process modeling [6,9], for workflow patterns [17], in the scientific workflow domain [11,13,26], and in the area of data integration [14,19]. Scientists using our approach select a few abstract patterns and combine them in their simulation workflows to describe only the main steps of these workflows. The adaptation to a concrete simulation scenario is achieved by a small set of pattern parameters that mainly correspond to terms or concepts scientists already know from their simulation models. Finally, rewrite rules transform such parameterized patterns into executable workflows. In this paper, we extend this vision of a pattern-based workflow design to a full-fledged and principled abstraction support for the complex data provisioning in simulation workflows. The main contributions are:

- We identify major requirements for the data provisioning in simulations.
- We derive a comprehensive set of patterns that may be used to alleviate the design of the data provisioning in several kinds of simulation workflows.
- We show how to organize patterns in a pattern hierarchy with different abstraction levels. As a major contribution, this pattern hierarchy facilitates a separation of concerns between different persons that are involved in workflow design, e.g., scientists, data management experts, or workflow engineers. According to his or her own skills, each person may choose the abstraction level s/he is familiar with and may provide other workflow developers with parameterized patterns and workflow fragments at this abstraction level.
- We discuss design considerations for a rule-based transformation of patterns into executable workflows and show its application to a concrete simulation.
- We evaluate the presented approach based on a prototypical implementation and based on its application to three real-world simulations.

In the remainder of this paper, we firstly detail our running example and exemplify major requirements for the data provisioning in simulations. In Section 3, we introduce the approach of pattern-based simulation workflow design and detail our set of patterns as well as the pattern hierarchy. The rule-based transformation of patterns into executable workflows is explained in Section 4. Section 5 covers information about our prototype and an evaluation of our approach. Finally, we discuss related work in Section 6 and conclude in Section 7.

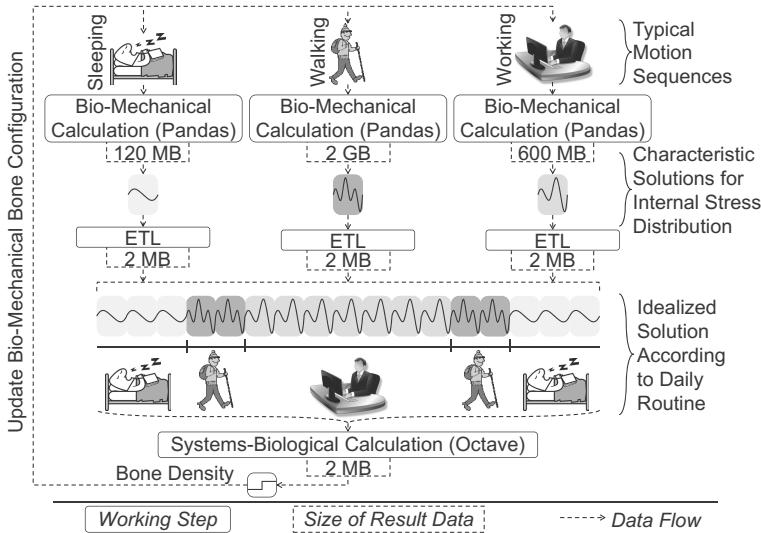


Fig. 1. Coupled simulation of structure changes in bones, cf. [8]

2 Data Complexity in Simulations

A solution to conquer the complexity of data provisioning in simulation workflow design has to consider a set of specific requirements. We introduce the discussion of these requirements by means of an example simulation.

2.1 Simulation of Structure Changes in Bones

As an example, we consider a simulation of time-dependent structure changes in bones, e. g., to support healing processes after bone fractures [8]. Fig. 1 shows how this simulation couples simulation models of the domains bio-mechanics and systems-biology, which involves lots of cooperative work across these domains. The bio-mechanical simulation model describes the mechanical behavior of a bone on a macroscopic tissue scale. However, it does not consider any cellular reactions within the bone tissue. This is where the systems-biological model comes into play, which determines the microscopic formation or resorption of the bone tissue as result of the stress-regulated interaction between cells.

The complexity of coupling simulations and providing the appropriate data for each of the simulation models is increased by the fact that typically separate simulation tools are employed for each of the models [8]. In our example, the Pandas framework¹, which is based on the Finite Element Method (FEM), offers a numeric implementation of the bio-mechanical simulation. The boundary conditions of this simulation correspond to the external load on the bone, e. g., caused by muscle forces. This external load depends on the motion sequence of

¹ <http://www.mechbau.uni-stuttgart.de/pandas/index.php>

the relevant person. Pandas converts the external load of each relevant motion sequence into a characteristic solution of the internal stress distribution within the bone tissue. For each numeric time step and for each nodal point of the FEM grid, it thereby stores up to 20 mathematical variables in a SQL database. This results in a data volume between 100 MB and several GBs per motion sequence.

The systems-biological calculation may be implemented via GNU Octave². It gets an idealized solution as input that is composed of the characteristic solutions of Pandas according to the approximated daily routine of the person. Beforehand, ETL processes filter appropriate data from the database of Pandas and aggregate these data among all numeric time steps, e.g., by calculating average values. Furthermore, they define the partitioning of the data among multiple instances of Octave to enable parallelism, and they store the results in CSV-based files (comma-separated values). Octave uses these files as input, calculates the precise bone density until the end of one daily routine, and stores it in another CSV-based file. This file is then imported into the database of Pandas to update the bone configuration of the bio-mechanical simulation model with the bone density. The whole process is repeated for the next daily routine.

2.2 Requirements for Data Provisioning

Such a complex data environment is common for simulations that are coupled among different scientific domains, since each domain has its own requirements and solutions for data handling. Due to the heterogeneity and high amount of involved data, scientists have to implement a multiplicity of complex data provisioning and data integration tasks. Workflows may help to structure these tasks [5], but they do not remove the burden from scientists to specify many low-level details of data management. To design an executable workflow realizing the simulation depicted in Fig. 1, scientists need to specify at least 22 complex workflow tasks [15]. This includes 9 low-level tasks that filter, aggregate, sort, split, and transform data. Here, scientists need to define sophisticated SQL, XPath, or XQuery statements. Scientists have much knowledge in their simulation domain, but rather limited skills regarding workflow or data management, and especially regarding SQL, XPath, or XQuery. So, they often waste time for workflow design they actually want to spend on their core issue, namely the simulation.

These issues imply a set of essential requirements that have to be met to ensure a wide adoption of simulation workflow technology. To identify these requirements, we have analyzed several academic use cases described in literature (see [20,24]), different workflow systems, and a set of real-world simulation processes. Beyond the simulation illustrated in Section 2.1, this set of real-word processes covers the examples described by Fehr et al. [3] and Rommel et al. [16]. In the following, we discuss the most relevant requirements:

- Requirements relevant for workflow design:
 - To conquer the complexity inherently associated with simulation workflows, we mainly require an *abstraction support for the definition of data*

² <http://www.gnu.org/software/octave/>

provisioning and data integration tasks that is suitable for scientists. On the one hand, this abstraction support should release scientists from defining complex implementation details or a high number of workflow tasks. On the other hand, it should enable them to work with terms or concepts they already know from their simulation models, instead of using workflow or data modeling languages they are not familiar with [11].

- The second requirement arises directly from the desire for coupling simulations and heterogeneous data environments of different scientific domains. To enable a seamless simulation coupling across arbitrary domains, an abstraction support has to be sufficiently *generic*, i.e., it has to consider all individual requirements of the respective domains [8,15].
- Requirements relevant for workflow execution:
 - The total size of the data involved in particular simulation runs ranges from a few MBs to several Terabytes. Furthermore, this may include a dynamically changing data volume, in particular for coupled simulations. Such a high and varying amount of data emphasizes the need for an *efficient data processing* and for *optimization opportunities* [2,12,21,25].
 - Scientists often make ad-hoc changes to workflows at runtime [22]. This requires an appropriate *monitoring* of workflow execution. Further relevant aspects are the reproducibility of a simulation and the traceability of its outcome, which has led to many *provenance* technologies [4].

In this paper, we propose a novel approach to solve the requirements regarding workflow design, which have largely been neglected in previous work. In the evaluation in Section 5, we nevertheless consider all four requirements.

3 Pattern-Based Simulation Workflow Design

In this section, we describe our approach that uses patterns to alleviate the design of simulation workflows. The core idea is that scientists select and combine only a few number of simulation-specific process patterns that represent high-level building blocks of simulation processes. These simulation-specific patterns combine several fine-grained workflow tasks, which reduces the number of tasks that are visible to scientists. Furthermore, they allow for a domain-specific and thus easy parameterization. After scientists have parameterized chosen patterns, these patterns need to be transformed into executable workflows. This is achieved by rewrite rules that recursively transform simulation-specific process patterns into more concrete workflow patterns, templates of workflow fragments, or services. Depending on their degree of implementation details, these workflow patterns, workflow fragments, and services may be provided by persons with different skills, e.g., by data management experts or workflow engineers. In the following subsections, we firstly discuss how to facilitate such a separation of concerns between different persons by organizing patterns in a pattern hierarchy with several abstraction levels. Afterwards, we illustrate the set of patterns and how they may be used for workflow design at each abstraction level. Information about the rule-based approach to pattern transformation are given in Section 4.

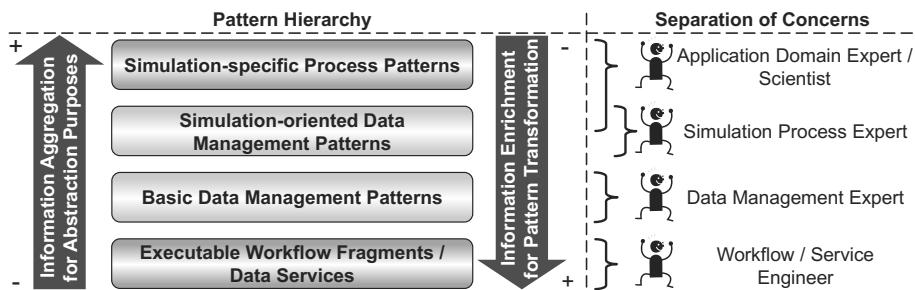


Fig. 2. Hierarchy of data management patterns with different abstraction levels

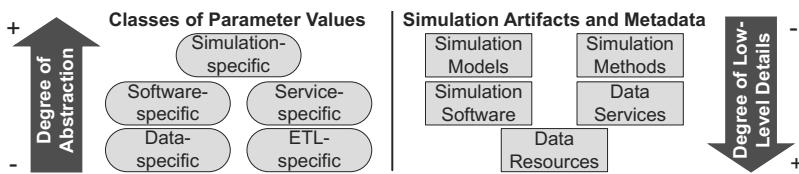


Fig. 3. Main classes and degree of abstraction of parameter values for patterns

3.1 Pattern Hierarchy and Separation of Concerns

Fig. 2 shows the afore-mentioned pattern hierarchy ranging from simulation-specific process patterns to executable workflow fragments and data services. To identify individual patterns and to properly arrange them in the hierarchy, we have analyzed the same uses cases as for the identification of requirements discussed in Section 2.2. In the following, we mainly focus on data management patterns that are relevant for data provisioning in simulation workflows.

As a major contribution, the clear distinction between individual abstraction levels in the pattern hierarchy enables a separation of concerns according to the skills of different persons involved in workflow design. Fig. 2 illustrates a possible separation of concerns between application domain experts (in our case scientists), simulation process experts, data management experts, and workflow or service engineers. According to his or her skills, each person may choose the abstraction levels s/he is familiar with. The person then may provide other persons with templates of parameterized patterns or workflow fragments at chosen levels. Thereby, the individual patterns serve as medium to communicate the requirements between different abstraction levels. For example, workflow or service engineers may offer executable workflow fragments or services at the lowest level of the hierarchy. In doing so, they only need to know the basic data management patterns provided by data management experts one level above, but they do not need to be aware of simulation-specific patterns at the two top levels.

With an ascending level of the pattern hierarchy, more information about data management operations and data management technology is aggregated. Hence, workflow developers need to know less about such operations and technology

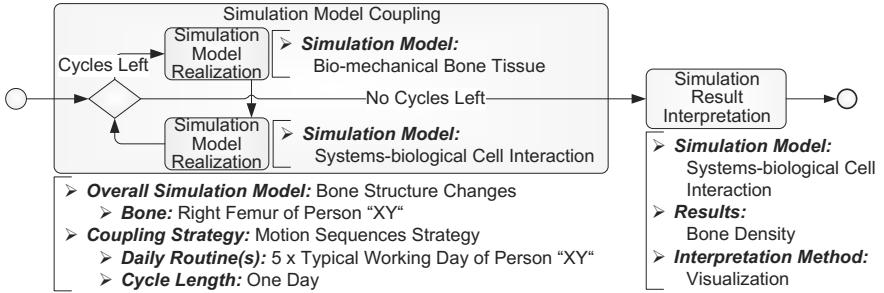


Fig. 4. Process model based on simulation-specific process patterns and their parameters for the simulation of structure changes in bones illustrated in Section 2.1

and may specify more abstract parameter values of patterns. Fig. 3 classifies parameter values and relates the resulting main classes according to their degree of abstraction, ranging from simulation-specific to data- or ETL-specific values. In line with current research efforts, we transfer the core idea of artifact-centric business process modeling [6,9] to simulation workflows and to corresponding simulation artifacts, such as mathematical simulation models or simulation software. Furthermore, we augment the artifact-centric idea by our pattern-based approach to workflow design. So, workflow developers may use simulation artifacts and their properties at different abstraction levels to specify parameter values of patterns. Additional metadata explicitly describe these simulation artifacts. This assists workflow developers in that they may choose values of some of the pattern parameters from the metadata. In the following, we detail patterns and their parameters at the individual levels of our pattern hierarchy.

3.2 Simulation-Specific Process Patterns

The top level of the pattern hierarchy shown in Fig. 2 comprises *simulation-specific process patterns* that focus on use cases scientists are interested in. Fig. 4 illustrates how such patterns may be used to describe the process of the bone simulation introduced in Section 2.1. The whole process consists of only four patterns that may be completely parameterized by *simulation-specific* values. These values correspond to artifacts scientists know from their domain-specific simulation methodology and are thus familiar with. Major examples of such artifacts are *simulation models* or mathematical variables of these models, as well as *simulation methods* such as the FEM. As such artifacts are often domain-specific, ontologies may be a good basis for metadata describing these artifacts [11,26]. Altogether, the strong relation to simulation-specific artifacts and to the use cases scientists are interested in makes simulation-specific process patterns good candidates to be selected and parameterized by scientists (see Fig. 2).

The main part of the process shown in Fig. 4 is a *Simulation Model Coupling Pattern* having the coupled simulation models for *bone structure changes* described in Section 2.1 as first parameter assigned. In addition, it defines the concrete *bone* to be simulated. The coupling pattern embeds one *Simulation Model*

Table 1. Simulation-oriented Data Management Patterns and their parameters

Pattern	Parameters ($< n..m >$ indicates cardinality)
<i>Simulation-oriented Data Provisioning</i>	<ul style="list-style-type: none"> – <i>Simulation model</i> $<1>$ – <i>Mathematical variables</i> $<1..n>$ – <i>Target</i> $<1>$: reference to software instance or service
<i>Simulation-oriented Data Interoperability</i>	<ul style="list-style-type: none"> – <i>Simulation models</i> $<2>$ – <i>Relationship between mathematical variables</i>: <ul style="list-style-type: none"> • <i>From first to second model</i> $<1..n>$ • <i>From second to first model</i> $<0..n>$ – <i>Partitioning mode</i> $<0..1>$
<i>Parameter Sweep</i>	<ul style="list-style-type: none"> – <i>Parameter List</i> $<1>$ – <i>Operation</i> $<1>$: workflow fragment, pattern, or service to be executed for each parameter in the list – <i>Completion Condition</i> $<0..1>$ – <i>Parallel</i> $<0..1>$: "yes" / "no", default is "no"

Realization Pattern for each of the two coupled simulation models, as well as the order in which they are executed. Such a pattern represents the complete realization of a simulation model, from initial data provisioning over calculations to result de-provisioning. The whole process executes the bio-mechanical model before the systems-biological one and repeats these executions for a certain number of coupling cycles. A further parameter of the overall coupling pattern completes this control flow definition by assigning the *coupling strategy* to be applied, e. g., the strategy illustrated in Fig. 1. The next parameter defines the *daily routines* and its composition of individual motion sequences. In addition, the *cycle length* determines the frequency in which the whole process re-iterates among both coupled simulation models. In our example, we consider a cycle length of one day and a total duration of five days, each corresponding to the typical working day of the relevant person. When all cycles, i. e., all days have been simulated, the process starts the result interpretation by means of a corresponding pattern. The pattern parameters specify which *results* of which *simulation model* shall be interpreted via which *interpretation method*, e. g., a visualization of the time-dependent bone density calculated by the systems-biological simulation.

3.3 Simulation-Oriented Data Management Patterns

Table 1 shows examples of *simulation-oriented data management patterns* that retain the abstraction level of simulation-specific values for most of their parameters. So, they may still be parameterized by scientists. Nevertheless, these patterns focus on use cases related to data management. As depicted in Fig. 2, this is where simulation process experts come into play who may assist scientists with orchestrating these patterns in their workflows. The *Simulation-oriented Data Provisioning Pattern* abstracts from data provisioning processes for simulation calculations or output visualizations, e. g., as part of the process patterns *Simulation Model Realization* or *Simulation Result Interpretation* shown in Fig. 4.

Table 2. Basic Data Management Patterns and their parameters

<i>Pattern</i>	<i>Parameters (<n..m> indicates cardinality)</i>
<i>Data Transfer and Transformation</i>	<ul style="list-style-type: none"> – <i>Sources <1..n></i>: references to data containers – <i>Targets <1..n></i>: references to data containers – <i>ETL processes for sources and targets <0..n></i>
<i>Data Iteration</i>	<ul style="list-style-type: none"> – <i>Data set <1></i>: one or a set of data containers – <i>Operation <1></i>: workflow fragment, pattern, or service to be executed for each relevant subset of the data set – <i>Resources <0..n></i>: e.g., references to data resources – <i>Completion Condition <0..1></i> – <i>Parallel <0..1></i>: "yes" / "no", default is "no" – <i>Data split <1></i>: e.g., data-specific partitioning mode or parameters of Data Transfer and Transformation Pattern – <i>Data merge <0..1></i>: similar to data split

The data to be provisioned is represented by a *simulation model* and by a set of its *mathematical variables*. In the example depicted in Fig. 1, the mathematical variables serving as input for the bio-mechanical simulation include its boundary conditions, e.g., the muscle forces. Only the *target* of the data provisioning needs to be specified via less abstract software- or service-specific parameter values, i. e., via a reference to a software instance or to a service that needs the data as input. Metadata describing software and services (and also data resources) as simulation artifacts may be based on common repository solutions.

The *Simulation-oriented Data Interoperability Pattern* defines data dependencies between *two simulation models* via simulation-specific parameter values [15]. This includes the *relationship between their mathematical variables*. Furthermore, a *partitioning mode* may define whether and how underlying data should be partitioned, e. g., to enable parallel executions of subsequent calculations. The *Parameter Sweep Pattern* supports processes that iterate over a *list of simulation-specific parameters* and execute an *operation* for each parameter in this list [12]. An optional *completion condition* defines whether and when the iteration should be finished before the whole parameter list is processed [7]. Another pattern parameter indicates whether the operation shall be executed in *parallel* or not.

3.4 Basic Data Management Patterns

On the way to executable workflows, these simulation-oriented patterns are intermediately mapped onto *basic data management patterns* whose parameters only may have service-, data-, or ETL-specific values. According to the separation of concerns depicted in Fig. 2, data management experts may use their knowledge to provide such less abstract parameter values or templates of them. Examples of basic data management patterns are listed in Table 2. The *Data Transfer and Transformation Pattern* may implement the Simulation-oriented Data Provisioning Pattern. The *sources* and *targets* of the corresponding data

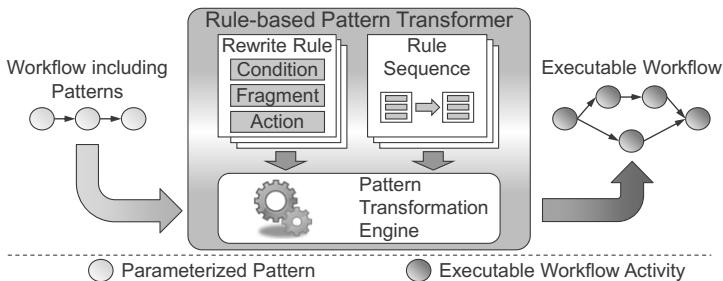


Fig. 5. Processing model of the rule-based transformation of patterns

provisioning process are specified as data-specific references to data containers, i.e., to identifiable collections of data, such as database tables or files. Furthermore, the pattern typically covers *ETL processes* including ETL operations to extract data sets from the source data containers, to transform these extracted data sets, and to load the transformation results into the target data containers.

The underlying principle of the *Data Iteration Pattern* is the iteration over a *data set* and the execution of an *operation*, where this data set serves as input. The operation may be executed on a set of *resources*, until an optional *completion condition* holds, and either in *parallel* or not. A *data split* parameter defines how to partition the data set among the resources, e.g., to enable parallelism based on MapReduce [1]. A possible value of this data split parameter is a data-specific partitioning mode, e.g., the equal distribution of tuples in a database table. As alternative, this data split may also be defined by means of parameters of a Data Transfer and Transformation Pattern. In a similar way, a *data merge* parameter may define how to integrate the results of the operation back into the data set. Altogether, this pattern may realize the Simulation-oriented Data Interoperability Pattern and the Parameter Sweep Pattern (see Section 4.2).

4 Rule-Based Pattern Transformation

The proposed pattern-based approach to simulation workflow design needs to be complemented by a strategy that transforms abstract patterns into executable workflows. In the following, we discuss basic design considerations of a rule-based transformation of patterns. Afterwards, we illustrate this pattern transformation for the simulation of structure changes in bones discussed in Section 2.1.

4.1 Basic Design Considerations of the Pattern Transformation

The processing model of the rule-based pattern transformation, shown in Fig. 5, extends some basic ideas of Vrhovnik et al. [25]. It traverses the graph of parameterized patterns given by the workflow. For each pattern, the *pattern transformation engine* tries to apply *rewrite rules*. Each rule includes a *condition* part that specifies the circumstances under which the remaining parts of the rule

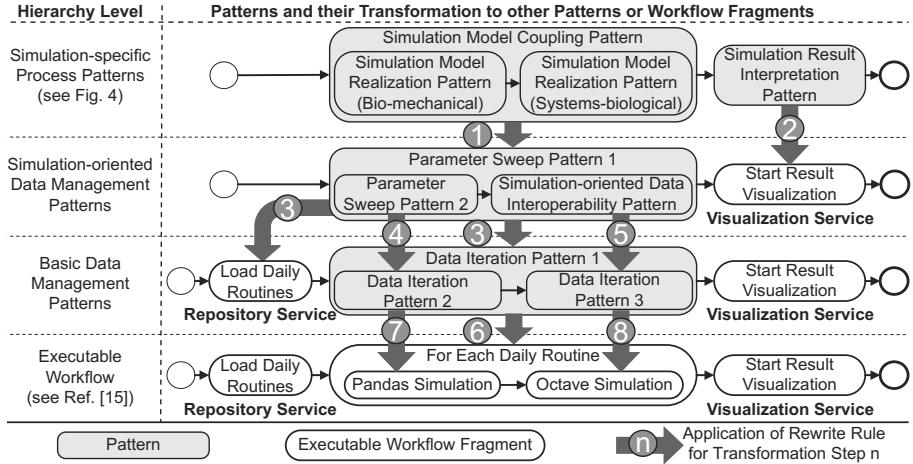


Fig. 6. Main transformation steps from the simulation-specific process model shown in Fig. 4 to the executable workflow illustrated by Reimann et al. [15]

may be applied to the pattern. The *fragment* part of the rule then identifies a template of a workflow fragment via a query to a workflow fragment library [18]. The *action* part defines transformation steps that add implementation details to the workflow fragment. Thereby, this action part may also map parameter values from high to low abstraction levels as shown in Fig. 3. Finally, the rule application replaces the pattern with the resulting workflow fragment. This workflow fragment may either be executable, which finishes the transformation of the pattern, or it may embed other patterns. In the latter case, the pattern transformer applies further rewrite rules to the workflow fragment and to its patterns until all final workflow fragments are executable. In addition, each level of the pattern hierarchy is associated with a *rule sequence* defining the set of rules that may be applied to corresponding patterns and the order in which these rules are tested for applicability. The first rule in a sequence whose condition part evaluates to true is applied to a pattern, while all remaining rules are neglected. Note that rewrite rules are not defined by scientists, but by other experts shown in Fig. 2, i.e., usually by those that also provide relevant workflow fragments.

4.2 Pattern Transformation in the Example Simulation

Now, we discuss how the pattern transformation is applied to the simulation-specific process model depicted in Fig. 4. Fig. 6 shows the main transformation steps over simulation-oriented data management patterns and basic patterns to the executable workflow. The workflow resulting from the simulation-specific process model after the transformation steps 1 and 2 consists of three simulation-oriented data management patterns and one service call. As a result of transformation step 1, the overall Parameter Sweep Pattern 1 implements the control flow of the coupling strategy that is specified as a parameter of the Simulation Model Coupling Pattern shown in Fig. 4. The simulation-specific parameter list

of the Parameter Sweep Pattern (see Table 1) defines the list of daily routines of the relevant person. The pattern sequentially iterates over this list and executes the two embedded patterns for each day. The first embedded pattern, i. e., Parameter Sweep Pattern 2, iterates in parallel over the list of motion sequences of the current day. The operation to be executed for each motion sequence is a service that implements the bio-mechanical Simulation Model Realization Pattern. The subsequent Simulation-oriented Data Interoperability Pattern abstracts from the data integration process required to couple the bio-mechanical and the systems-biological simulation models. It therefore defines the relationships between their mathematical variables as described by Reimann et al. [15]. Some patterns may also be directly mapped to an executable workflow fragment or service without traversing the whole pattern hierarchy depicted in Fig. 2. In our example, this is the case for the Simulation Result Interpretation Pattern, which is mapped to the afore-mentioned final service call (transformation step 2 in Fig. 6).

After transformation steps 3 to 5, the workflow consists of three basic data management patterns and two service calls as shown in Fig. 6. At this abstraction level, the patterns are specified mainly via data-, service-, and ETL-specific parameter values. Finally, rewrite rules of the last transformation steps 6 to 8 transform these basic patterns into executable workflow fragments to create the executable workflow illustrated by Reimann et al. [15].

5 Evaluation and Discussion

In this section, we discuss the results of our evaluation of the pattern-based approach to simulation workflow design. This evaluation has been backed by a prototypical implementation, which we have applied to three real-world simulations. This includes the bone simulation introduced in Section 2.1, as well as the examples described by Fehr et al. [3] and Rommel et al. [16]. The prototype is based on the workflow language BPEL [7], on the workflow design tool Eclipse BPEL Designer³ Version 0.8.0, and on the workflow execution engine Apache Orchestration Director Engine⁴ (ODE) Version 1.3.5. We use a PostgreSQL⁵ Version 9.2 database system to store the metadata of simulation artifacts depicted in Fig. 3. For the rule-based pattern transformation depicted in Fig. 5, we have extended the JRuleEngine⁶ to support rewrite rules and rule sequences for the three example simulations stated above. In the following, we evaluate to what extent our approach fulfills the requirements regarding workflow design and the requirements regarding workflow execution discussed in Section 2.2.

5.1 Assessment of the Abstraction Support

Fig. 7a shows the number of workflow tasks to be specified for the bone simulation introduced in Section 2.1 at the different abstraction levels depicted in

³ <http://www.eclipse.org/bpel/>

⁴ <http://ode.apache.org/>

⁵ <http://www.postgresql.org/>

⁶ <http://jruleengine.sourceforge.net/>

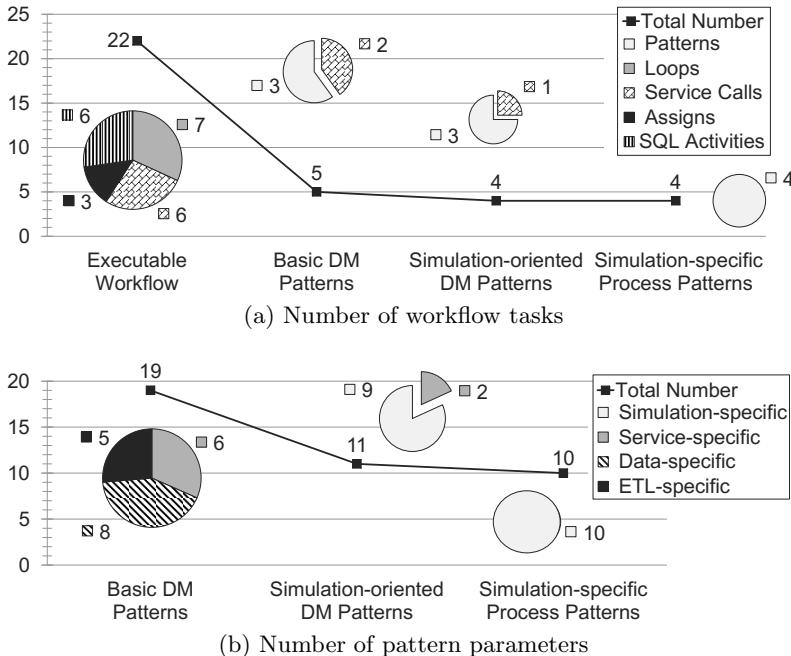


Fig. 7. No. of workflow tasks and parameters to be specified for the example workflow at different abstraction levels; The partitioning among different types of workflow tasks or parameters is shown in pie charts; "DM" means "data management"

Fig. 6. Furthermore, pie charts in Fig. 7a illustrate the partitioning of these workflow tasks among abstract *patterns* and among different types of executable tasks. With an increasing design complexity, the executable tasks range from simple *loop* structures over *service calls* to complex BPEL *assigns* or *SQL activities* [25]. BPEL assigns implement complex XPath or XQuery expressions, while SQL activities cover even more sophisticated SQL statements [15]. If scientists were designing the executable simulation workflow at the lowest abstraction level as illustrated by Reimann et al. [15], they would need to define a total number of 22 workflow tasks and many complex implementation details. In particular, this would include 9 sophisticated tasks for assigns and SQL activities with complex XPath, XQuery, or SQL statements. Since scientists are typically not familiar with defining such low-level statements, they would not accept this huge design effort. As shown in Fig. 7a, the three pattern abstraction levels depicted in Fig. 6 not only reduce the complexity of workflow tasks, but also their total number to 5 and 4, respectively. This constitutes a total reduction by a factor of 5.5.

We have furthermore analyzed the number and complexity of pattern parameters to be specified at each of the three pattern abstraction levels. Fig. 7b shows these numbers of pattern parameters and illustrates their partitioning according to their complexity, i. e., according to the parameter classes given in Fig. 3. With an increasing abstraction level, the total number of pattern parameters

is nearly divided in half. The basic data management patterns mainly require data- and ETL-specific parameter values, namely 13 of all 19. The other abstraction levels however completely neglect these complex parameter classes. In fact, they mainly consider simulation-specific parameter values that abstract from any implementation details and that are particularly suitable for scientists.

With respect to the other simulation examples described by Fehr et al. [3] and Rommel et al. [16], we have gained nearly similar results as reported in Fig. 7a and 7b. The reduced number and complexity of both workflow tasks and pattern parameters represent a considerable simplification for simulation workflow design. The distribution of these numbers and of the complexity among individual abstraction levels also complies with the skills of different persons involved in workflow design, as it is depicted in Fig. 2. This finally shows the suitability of the separation of concerns introduced by our pattern hierarchy.

5.2 Generic Data Management in Workflows

We also used our prototype to asses whether our patterns are generic enough to cover all relevant data provisioning steps in different simulations [3,16]. In the example described by Rommel et al. [16], we have been able to use the different *simulation-specific process patterns* depicted in Fig. 4 to describe the whole simulation process. This is because these patterns cover common use cases in simulations. In a similar way as shown in Fig. 6, we also mapped them to the Parameter Sweep Pattern and Simulation-oriented Data Interoperability Pattern, as well as to the Data Iteration Pattern. Model reduction, i. e., the reduction of the number of degrees of freedom in a simulation model to reduce computational costs, is another important issue for simulations [3]. This is one of the rare use cases that benefits from a new simulation-specific process pattern incorporating domain-specific parameters. These pattern parameters are (1) the simulation model to be reduced, (2) the concrete reduction method to be employed, (3) the desired number of degrees of freedom, and (4) the required quality of the reduced model. The patterns at lower hierarchy levels nevertheless consider more generic parameters that are independent of the concrete simulation domain. Using our prototype, we again mapped the domain-specific model reduction pattern to the Parameter Sweep Pattern and subsequently to the Data Iteration Pattern.

5.3 Efficient Data Processing and Optimization

The rule-based pattern transformation may cause an overhead when applied at workflow runtime. We have carried out experiments to evaluate this overhead. Our prototype ran on a 64 bit system with 32 GB RAM and an Intel Xeon CPU with 8 cores. We have run the pattern transformation for the 8 transformation steps depicted in Fig. 6 100 times and have determined its average duration. To assume a worst-case scenario, our implementation of the pattern transformation checks 100 rules for each transformation step before the 100th rule is applied. The resulting worst-case duration of the pattern transformation is 0.36 seconds. We have also measured the duration of the executable workflow for the simulation of

Table 3. Overhead of pattern transformation for its worst-case duration of 0.36 seconds

<i>Number of tuples</i>	1 million	10 million	100 million
<i>Workflow duration</i>	140 seconds	1410 seconds	14095 seconds
<i>Worst-case overhead</i>	0.26 %	0.026 %	0.0026 %

structure changes in bones [15]. The workflow and the simulation services it calls ran in parallel on seven computers, each equipped with the hardware resources described above. We have executed the workflow 10 times and for a varying number of tuples as Pandas typically stores them in its output database table, i. e., 1, 10, and 100 million tuples. Table 3 illustrates the respective average workflow durations. Furthermore, it shows the overheads caused by the pattern transformation for its worst-case duration of 0.36 seconds. The maximum overhead is 0.26 % and thus negligible compared to the workflow duration.

The rule-based approach for pattern transformation furthermore enables a seamless integration of rule-based optimization decisions [12,25]. When we increase the number of tuples Pandas stores in its database to 1 billion, our measurements reveal that the workflow realizing the simulation of structure changes in bones [15] lasts about six hours. The ETL processes that are depicted in Fig. 1 and that are part of this workflow may be executed independently for each nodal point of the FEM grid and thereby drastically reduce the data sizes. This makes these ETL processes good candidates to be realized via MapReduce [1,10]. Rewrite rules may choose such more efficient realizations at workflow runtime.

5.4 Monitoring and Provenance Support

While our approach reduces the number and complexity of workflow tasks that are visible to scientists, this may cause a problem for monitoring and provenance. The workflow execution environment is only aware of the more complex executable workflows resulting from the pattern transformation. This may lead to a missing correlation between patterns visible in a workflow design component and audit or provenance information captured by an execution environment. Scientists might be confused when they monitor workflow executions or try to reproduce simulations. A possible solution to this problem could be a view concept on workflows [23]. After each application of a rewrite rule in the pattern transformation, the input pattern is not replaced by the resulting workflow fragment, but it is defined as view on this fragment. Such views enable an aggregation of audit or provenance information to recover their correlation to individual patterns [23]. Furthermore, views also allow scientists to look into specific low-level details of data provisioning tasks if this is required for monitoring purposes.

6 Related Work

According to our focus on data provisioning in workflows, we now discuss related work for data- and artifact-centric business process modeling, for workflow patterns, as well as from the areas of scientific workflows and data integration.

Artifact-centric approaches to business process modeling leverage data as first-class citizens in processes [6,9]. The data is represented by so-called business artifacts that correspond to business-relevant objects, e.g., a customer order or an invoice. The artifacts manage relevant information about these business objects and about their life cycle, and they control how services may be invoked on the objects. As described in Section 3.1, we transfer the core idea of business artifacts to simulation workflow design and to simulation artifacts, e.g., mathematical simulation models. In addition, we augment the artifact-centric idea by our pattern-based approach to workflow design and by the separation of concerns with clearly distinguished abstraction levels as depicted in Fig. 2. From the perspective of scientists, the patterns significantly reduce both the number and complexity of workflow tasks. This is an important issue for the acceptability of simulation workflow technology. Altogether, the combination of an artifact-centric approach with explicitly described patterns of typical workflow tasks considerably improves the way of how to design simulation workflows.

Russel et al. discuss common workflow data patterns [17]. The authors primarily consider fine-grained patterns serving as benchmark to evaluate and compare different workflow languages or workflow systems. For instance, some of the patterns deal with the question whether workflow activities may exchange data by value or by reference. Due to their less technical background, scientists would typically not accept such fine-grained patterns as building blocks for simulation workflow design. Instead, these patterns rather classify implementation details of executable workflow fragments at the lowest level of our pattern hierarchy.

Related work from the scientific workflow domain makes use of ontologies to allow for an abstract parameterization of scientific workflow tasks, i.e., the parameters of these tasks correspond to domain-specific terms or concepts known from the ontologies [11,26]. Furthermore, logical rules may define mappings of such abstract workflow tasks or ontology terms to executable workflows, services, or low-level data types [11,13]. However, the mentioned approaches only moderately reduce the number of workflow tasks that are visible to scientists, while our patterns significantly reduce both the number and the complexity of these tasks. Furthermore, the sole use of ontologies often entails that these approaches are restricted to certain application domains, e.g., life sciences or geophysics. In contrast and as discussed in Section 5.2, our patterns are generic enough to be applied in various domains of simulation applications.

Federated information systems integrate diverse data sources and provide a uniform data schema and a uniform query language [19]. Such uniform schemas and languages however still do not remove the burden from scientists to specify low-level details in terms of complex queries, e.g., based on SQL or XQuery. Furthermore, federated schemas often lack the generality needed to support different scientific domains. Common ETL tools or approaches to schema mapping and matching may help to define ETL processes in basic data management patterns [14]. However and as discussed in Section 3, scientists rather prefer simulation-oriented data management patterns or even simulation-specific process patterns that do not explicitly consider ETL processes.

7 Conclusion and Outlook

Simulations often involve multiple, highly complex data provisioning and data integration tasks, particularly when they are coupled among different scientific domains. In order that scientists are able to concentrate on their core issue, namely on simulation modeling, an abstraction support for this complex data management is indispensable. In this paper, we have presented a novel approach for a pattern-based design of the data provisioning in simulation workflows that conquers the data complexity in such workflows. A pattern hierarchy with different abstraction levels enables a separation of concerns according to the skills of different persons involved in workflow design. For example, scientists select and parameterize a few abstract, simulation-specific process patterns to only describe the core aspects of simulation processes. We use a transformation approach based on a set of rewrite rules that map such abstract process models and patterns onto executable simulation workflows. A prototypical implementation and its application to several simulation applications, e.g., to bio-mechanical and systems-biological problems, has served as basis to evaluate this approach. The patterns significantly reduce both the number and the complexity of workflow tasks that are visible to scientists. The generality of patterns furthermore enables their seamless application in various simulation domains.

As part of future work, we will even increase this generality by working on additional patterns and by applying our approach to other application domains than simulations, e.g., to business processes. To further evaluate the abstraction support and user satisfaction offered by our approach, we will conduct usability studies with real scientists. Furthermore, we will investigate how to integrate optimization decisions into rewrite rules to increase the efficiency of workflows.

Acknowledgments. The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1). We also thank our colleagues Dimka Karastoyanova and Frank Leymann for their valuable cooperation.

References

1. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM* 51(1) (2008)
2. Deelman, E., et al.: Pegasus: A Framework for Mapping Complex Scientific Workflows Onto Distributed Systems. *Scientific Programming* 13(3) (2005)
3. Fehr, J., et al.: Simulation Process of Flexible Multibody Systems with Non-modal Model Order Reduction Techniques. *Multibody System Dynamics* 25(3) (2011)
4. Freire, J., et al.: Provenance for Computational Tasks: A Survey. *Computing in Science and Engineering* 10(3) (2008)
5. Görlich, K., et al.: Conventional Workflow Technology for Scientific Simulation. In: *Guide to e-Science*. Springer, London (2011)
6. Hull, R.: Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: Proc. of the 7th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE), Monterrey, Mexico (2008)
7. Jordan, D., Evdemon, J.: Web Services Business Process Execution Language Version 2.0, OASIS Standard (2007)

8. Krause, R., et al.: Scientific Workflows for Bone Remodelling Simulations. *Applied Mathematics and Mechanics* 13(1) (2013)
9. Künzle, V., Reichert, M.: PHILharmonicFlows: Towards a Framework for Object-aware Process Management. *Journal of Software Maintenance and Evolution: Research and Practice* 23(4) (2011)
10. Liu, X., Thomsen, C., Pedersen, T.B.: ETLMR: A Highly Scalable Dimensional ETL Framework Based on MapReduce. In: Cuzzocrea, A., Dayal, U. (eds.) *DaWaK 2011*. LNCS, vol. 6862, pp. 96–111. Springer, Heidelberg (2011)
11. Ludäscher, B., Altintas, I., Gupta, A.: Compiling Abstract Scientific Workflows into Web Service Workflows. In: Proc. of the 15th International Conference on Scientific and Statistical Database Management, Cambridge, MA, USA (2003)
12. Ogasawara, E.S., et al.: An Algebraic Approach for Data-Centric Scientific Workflows. In: Proc. of the 37th International Conference on Very Large Data Bases (VLDB 2011), Seattle, WA (2011)
13. Radetzki, U., et al.: Adapters, Shims, and Glue – Service Interoperability for in Silico Experiments. *Bioinformatics* 22(9) (2006)
14. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. *International Journal on Very Large Data Bases (VLDB Journal)* 10(4) (2001)
15. Reimann, P., Schwarz, H., Mitschang, B.: Data Patterns to Alleviate the Design of Scientific Workflows Exemplified by a Bone Simulation. In: Proc. of the 26th International Conference on Scientific and Statistical Database Management (2014)
16. Rommel, J.B., Kästner, J.: The Fragmentation-Recombination Mechanism of the Enzyme Glutamate Mutase Studied by QM/MM Simulations. *Journal of the American Chemical Society* 26(133) (2011)
17. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow Data Patterns: Identification, Representation and Tool Support. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) *ER 2005*. LNCS, vol. 3716, pp. 353–368. Springer, Heidelberg (2005)
18. Schumm, D., et al.: Process Fragment Libraries for Easier and Faster Development of Process-based Applications. *Systems Integration* 2(1) (2011)
19. Sheth, A.P.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. In: Proc. of the 17th International Conference on Very Large Data Bases (VLDB 1991), Barcelona, Spain (1991)
20. Shoshani, A., Rotem, D.: Scientific Data Management: Challenges, Technology, and Deployment. Computational Science Series. Chapman & Hall (2009)
21. Simitis, A., et al.: Optimizing Analytic Data Flows for Multiple Execution Engines. In: Proc. of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD 2012), Scottsdale, AZ, USA (2012)
22. Sonntag, M., Karastoyanova, D.: Next Generation Interactive Scientific Experimenting Based on the Workflow Technology. In: Proc. of the 21st IASTED International Conference on Modelling and Simulation, Prague, Czech Republic (2010)
23. Sonntag, M., et al.: Views on Scientific Workflows. In: Proc. of the 10th International Conference on Perspectives in Business Informatics Research (2011)
24. Taylor, I., Deelman, E., Gannon, D.: Workflows for e-Science - Scientific Workflows for Grids. Springer, London (2007)
25. Vrhovnik, M., et al.: An Approach to Optimize Data Processing in Business Processes. In: Proc. of the 33rd International Conference on Very Large Data Bases (VLDB 2007), Vienna, Austria (2007)
26. Wolstencroft, K., et al.: The myGrid Ontology: Bioinformatics Service Discovery. *Int. Journal on Bioinformatics Research and Applications* 3(3) (2007)

Augmenting and Assisting Model Elicitation Tasks with 3D Virtual World Context Metadata

Ross Brown¹, Stefanie Rinderle-Ma²,
Simone Kriglstein³, and Sonja Kabicher-Fuchs⁴

¹ Queensland University of Technology, Information Systems School, Australia
r.brown@qut.edu.au

² University of Vienna, Faculty of Computer Science, Austria
stefanie.rinderle-ma@univie.ac.at

³ Vienna University of Technology, Human-Computer Interaction Group, Austria
kriglstein@cvast.tuwien.ac.at

⁴ Inset Advisory, Austria
sonja.kabicher-fuchs@Inset-advisory.com

Abstract. Accurate process model elicitation continues to be a time consuming task, requiring skill on the part of the interviewer to process information from interviewees. Many errors occur in this stage that would be avoided by better activity recall, more consistent specification and greater engagement by interviewees. Situated cognition theory indicates that 3D representations of real work environments engage and prime viewer cognitive states. In this paper, we augment a previous process elicitation methodology with virtual world context metadata, drawn from a 3D simulation of the workplace. We present a conceptual and formal approach for representing this contextual metadata, integrated into a process similarity measure that provides hints for the business analyst to use in process modelling steps. Finally, we conclude with examples from two use cases to illustrate the potential abilities of this approach.

Keywords: Process Elicitation, Process Context Metadata, Human-centric BPM, Process Modelling, 3D Virtual Worlds.

1 Introduction

Process model elicitation still poses a huge challenge with respect to the quality of the resulting process models, independently of whether the information was gathered from interviews [13], by exploiting existing data sources [6], or by process mining [3].

We will refer to two selected challenges, i.e., *imprecise activity names* and *imprecise time stamps* as described in [3]. Imprecise activity names occur, for example, if interviewees describe their work tasks at different granularity levels. Within a case study from the higher education domain [13], this problem became immediately evident. Also in a case study from the health care domain described in [6], analysis of existing data sources revealed description and storage of activities at different granularity levels. Finally, the analysis presented in

[3], underpinned the existence of this problem in process logs, e.g., from Business Process Intelligence (BPI) challenges. Imprecise activity names might lead to quality problems with resulting process models, as activities that actually describe the same work task might be identified as different activities and vice versa, leading to overly complex and possibly incorrect process models. Imprecise time specifications occur frequently as well and have a direct impact on the activity sequencing [6]. When interviewing process participants as described in [13], they may not directly talk about the order of their activities and most probably not provide any time stamps. Existing data sources often contain imprecise time stamps due to logging at different granularity levels. A prevalent problem then is that activities might be detected as happening in parallel, e.g., on the same day, despite them having a serial order within the day. Some solutions to tackle such problems have been presented in [6] by enriching the process model knowledge with metadata. This metadata may be derived from the work environment, or it may have to be elicited from process stakeholders in an a posteriori manner[2].

From this thinking, the following three research questions arise:

- R1 – what additional information can be exploited in order to overcome challenges around imprecise activity names and imprecise time stamps?
- R2 – how can we elicit this information?
- R3 – how can we measure the success of eliciting and exploiting additional information?

Addressing R1, in this paper, we analyze the approach presented in [13] for eliciting process model information from interviews. Specifically, [13] provides a list of so called *hints* that were typically mentioned by the interviewees. As a conceptual contribution of this paper, it will be shown how the information from the hint list can be mapped onto virtual world contextual information connected with process activities.

For R2, we posit that an enhanced memory model needs to be used to provide the prompting required during interviewing approaches. What is needed is a methodology to enhance the interview process to provide information necessary to provoke detailed responses from the stakeholder. Such an enhanced memory model can also be justified from a cognitive psychological point of view [10], whereby the presentation of visual stimuli enables the accessing of longer term memory within the brain that provides more information for the stakeholder to use. The argument is that personally situated work representations can increase the loading capability of the short-term memory and arouse prior knowledge in the longer-term memory, consistent with the arguments of Miller [18]. For example, an accurate visualization of a working environment can arouse the prior knowledge of a stakeholder. A personalised visualization of their working environment can provide them with a representation in a “hands-on” manner that simulates real artefacts in real spaces. This enables them to make comments according to their deeper work experiences. Moody concludes that this “hands-on” manner is a form of semantic transparency [19]. People can directly infer information for a conceptual model from the operational representation of the conceptual models in a simulated workplace, in a manner that is consistent with

models of situated cognition, that is, that we store work process memories as situated actions, relating our work to physical resources in our workplace [5]. Therefore, we believe it can be argued that an interactive 3D representation of a workplace will provide these stimuli [10]. This is key to the interviewing process that is performed at the commencement of process modelling with stakeholders. Assuming that a work situated, observational version of interviewing is canonical (but not necessarily possible) we suggest that an interactive virtual world will be a major step forward in providing a similar set of stimuli to provoke these memories of a process, thus providing personalised to-do lists that may be merged into a final overall process model for a number of roles.

Finally, we will address R3 by evaluating the approach based on two case studies 1) follows up on previous work by the authors [13] in the context of higher education processes at the University of Vienna and 2) takes logs from the BPI 2011 challenge [16] containing data from the health care domain.

The Vienna case study continues our previous work, applying our new approach to extract the process model of teaching from to-do's of selected process participants at the Faculty of Computer Science, University of Vienna. In face-to-face interviews, selected people working in key positions of the teaching process were asked to list their activities in the process under investigation, similar to a scheduled to-do list. We use examples from this case study to highlight the extra spatial information that can be drawn from the virtual world information that can be used in such circumstances. In particular, we show how to use this information to reinforce the previously mentioned hint process to provide greater insights into the specified activities. For each extracted case, logs previously garnered from interviews were examined for contextual information that could provide hints as to new structures within the final process model.

To round out the content of the case studies, we have included from a separate study two examples from the Business Process Intelligence (BPI) 2011 challenge [16]. The BPI challenge is an annual process mining competition, used to test, compare and improve process log analysis techniques. The logs are raw data drawn from the Academisch Medisch Centrum (AMC) hospital, a teaching hospital in Amsterdam, the Netherlands. Diagnostic and treatment activities recorded in the logs were examined from a group of 627 gynaecological oncology patients treated in 2005 and 2006.

In each case there are issues with ambiguity and lack of precision in the specification of log activities. In both cases, we aim to show that the original processes, without using the virtual world and exploiting hints, can be enhanced in quality by using the virtual world metadata. In addition, the two case studies support our argument that the new approach is easy to apply to widely differing process modelling domains, such as education and health.

The structure and contributions of the paper are as follows. Section 2 is a theoretical mapping of key contextual information to hints for use by the process modeller. We then show how these hints relate to key challenges in process log quality used in process mining. Section 3 describes the design and implementation of a novel virtual world interview tool to help elicit process activities.

From this initial analysis and development, a formal set of operators has been developed to apply to the resulting contextual data in Section 4. Next, an initial theoretical analysis in Section 5 is applied to an admixture of real and synthetic use case data, illustrating the ability of the hint in guiding modelling decisions. The paper concludes with a listing of relevant work in the field and discusses future work.

2 BPMEVW: Virtual World Extended BPME

In this paper, we present a 3D virtual world process model elicitation approach for the purpose of assisting with process elicitation by supporting better personal activity recall and by providing hints for later analysis tasks being performed. The Business Process Model Extraction (BPME) method as presented in [13] can be extended by using a virtual world as the first stage in the individual view elicitation. The newly modified methodology – called Virtual World Extended BPME (BPMEVW) – is depicted in Figure 1, with a new stage involving the use of a 3D virtual world tool in the process elicitation phase.

We should note that we have chosen for a number of reasons to use the BPME method, extended with a virtual world into the BPMEVW. The BPME hint structure conceptually aligns with the contextual priming provided by the virtual world representations. In addition, virtual world context metadata can be applied to stages in the process of creating process models in the BPMEVW, allowing a context information model to support further process modelling steps, see the *World Context* document in Figure 1. The BPMEVW method also supports a transparent and comprehensible way of developing process models. Typical methods that aim at the elicitation of process information are, for example, the scanning of textual process descriptions, if available, by the business analyst, interviews performed with persons acting in specific process-relevant roles in the enterprise and workshops in which processes are described and modelled in groups. Although all these elicitation methods have their abilities, often there might arise a kind of black box between the process elicitation and the final model. The virtual world helps us to elicit process information from the user, and at the same time, automatically document the information in terms of a process elicitation log. In later steps, the process elicitation log is prepared and used automatically for process mining to obtain the final model, thus offering a fully documented, comprehensible and repeatable process of process modelling.

From such an approach, we propose that the augmented virtual world method will provide the following hints to the BMPEVW approach. Each of these may be articulated with extra context information that provides insight into process activity equivalence. In order to provide answers to R1 (see Introduction), we analyse each in turn for insight into the ability of the context information to refine this process.

Hint 1 - data transfer - (datainput and dataoutput) or events (e.g., eMail, Mail, and PhoneCall) between two or more process participants (connection). For example, to-do of the AdministrativeCooperator: “Mail lecturing contract to

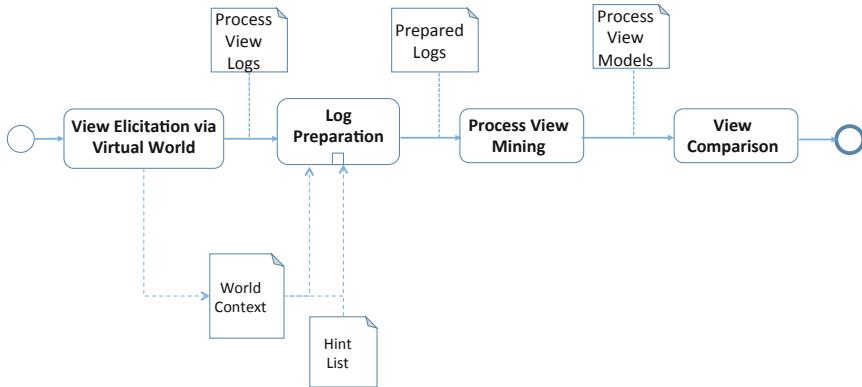


Fig. 1. BPMEVW method, augmented with virtual world-based process elicitation activities and associated metadata to assist with merging and alignment activities.

lecturer,” and to-do of a Lecturer: “Receipt of the lecturing contract in my post office box.” In the virtual world, human resources can be represented visually, thus direct interactions with people can be recorded automatically in the world, providing an explicit mechanism to record human to human data interactions.

Hint 2 - group tasks and activities (participation) - for example, to-do of the Lecturers and the TeachingCoordinators: “Participation in coordination meeting.” This can be determined by extracted world location information. If activities occur within a particular location, then an analysis of the 3D space with different subjects will indicate a potential group activity.

Hint 3 - decisions - for example, to-do of a Lecturer: “Either I contact the AdministrativeCoordinator if I want to book the lecture hall X or I contact the Secretary if I want to book the lab.” While explicit choices cannot be shown in the world we have implemented, there is definitely the ability to record the *if* as a choice or decision by recording two instances of an activity with an embedded *if* in the text.

Hint 4 - delegation - it can be assumed that the to-do’s are performed by the process member who listed the to-do’s. Otherwise, a hint to another person is given. For example, to-do of a ModuleCoordinator: “A lecturer of our lecture team books a lecture hall for all of us”. If the activity has another human resource involved, then this is a strong hint of a delegation being performed.

Hint 5 - tools - it might be the case that process participants would rather mention the particular tool name (e.g. Fronter) than the general term (e.g. learning platform). For example, to-do of a Lecturer: “Then I enter the grades of the students into ISWI¹. ” The interviewee may interact with a mobile phone to define an activity, either as: phone person to book room, or send email to person to book room. Indeed, we conjecture that specifying the object used to perform the task will provide a priming prompt to divulge the tools used.

¹ Online platform at the Faculty of Computer Science, University of Vienna

Hint 6 - reoccurring activities - for example, to-do of a Lecturer: “Reoccurring task: conduction of the units”. The virtual world accommodates the definition of repetitious activities via providing extra information to correlate with the activity label, for example, the location and resources used in the task will provide further hints that these repeating activities are indeed the same task being repeated.

Hint 7 - time notes - we assume that the to-do list already reflects a sequential order of the to-do’s. The first to-do mentioned in the list is the first to-do that is performed by the person in the process under investigation. Explicitly mentioned time notes may be vague but support the designer to sequentially order summarised tasks and activities. For example, to-do of a Lecturer: “At the day of the unit I print the attendance list.”, or “Before the semester starts I plan the course”. Such a hint addresses issues raised by [3], instead of the interviewee relying on their memory, the priming of the world will push the user towards a temporally aligned specification of the to-do list.

Hint 8 - activity merge - in addition to the seven hints listed by [13], we add an activity merging hint, drawn from the context metadata provided within the virtual world. In essence, along with simple activity string similarity measures, similarity measures can process extra virtual world context metadata, for example, resources (human and non-human) and locations.

3 Virtual World Representation

We use 3D virtual world technology to provide the interactive environment to support process elicitation tasks. Virtual worlds are immersive 3D environments which contain 3D artefacts and avatars that have real-time interaction capabilities [4]. Such technology is ubiquitous, due to the rise of advanced graphics technology, enabling virtual world systems to run on standard desktops. The focus of the representation component is thus not the virtual world technology, but the actual capabilities and realism required for the visual representation to be used. Major open source components of the implementation include a world server (OpenSim²) a viewing client (Firestorm³), a MySQL database backend (WAMP⁴) and some small PHP applications to generate .xes format process logs.

Using a floor plan and imported artefacts, the general layout of key areas can be quickly modeled for use in an interview scenario. For our two use cases, we analysed typical workspaces to find out what are the key objects in the rooms (e.g., chairs, tables and phones) in order to select corresponding virtual world artefacts. A key research question is the level of detail required for the visualization of the workplace. A complete model is out of the scope of this paper, however, it should be noted that the visual fidelity of the environment is modulated by the interaction needs of the space [15]. A designer will provide

² Opensimulator: www.opensimulator.org, accessed: June, 2014

³ Firestorm: www.firestormviewer.org, accessed: June, 2014

⁴ WAMP: www.wampserver.com, accessed: June, 2014

artefacts in the environment to support the extraction of relevant events for process mining. Detail is focused upon the items that are specifically used in processes. The more superfluous items, with regards to this simulation, are the walls and roofs of the buildings, as they do not play as strong a part in the process of priming people [10]. We argue logically that the objects of most relevant affordance should be presented at the highest level of detail as they have the most influence on the cognition of the user with respect to their process activities. We argue that the rest of the scene may be left in lower levels of detail, with a lowered effect on the tasks being elicited. As a corollary, we have adopted a common visualization approach of presenting our examples with floor plans and uploaded meshes from free sources (3D Sketchup Warehouse⁵). Examples of the environment have been developed for our case studies and are shown below in Figure 2. We used office artefacts gained from the Google 3D SketchUp service, thus no real modelling was required for the office example. In the hospital version, the objects in the scene have been constructed by hand, but similarly, representational hospital objects can be found in the online Sketchup database. We emphasise that, in these use cases, low amounts of modelling were required in order to generate a useful virtual world.



Fig. 2. Images from virtual worlds developed for the Vienna Computer Science (left image) and Dutch hospital (right image) case studies.

3.1 Interaction Approach

As this project is, at its essence, a process elicitation project, we have sought to integrate methods for process activity definition in a manner similar to free form interview answers. We seek to provide a solution for R2 (see Introduction) by designing an elicitation method that intuitively incorporates a 3D virtual world view. The 3D view is from a worker perspective, being an avatar-based third person view [4]. This world view provides a setting that is cognitively subjective, facilitating a personal viewpoint when eliciting to-do lists from interviewees. An

⁵ 3D Sketchup Warehouse: 3dwarehouse.sketchup.com, accessed: June, 2014

interaction approach was chosen in order to configure the virtual world client and server for optimal use by interviewees. Therefore, the basis for interaction is a free form specification of text describing executed activities. This is motivated by a number of factors. Firstly, when being interviewed by a business analyst, the responses of the interviewee are free form text as speech or questionnaire responses [13]. In principle, we provide an interaction mechanism that is as close to the data capturing done by an interviewer as possible, see Figure 3.

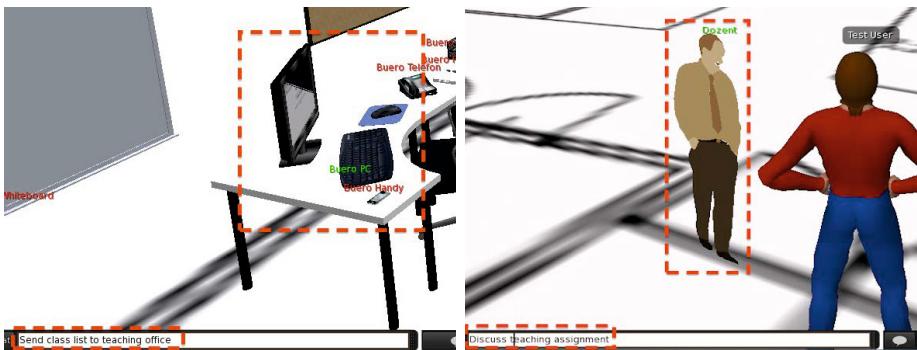


Fig. 3. Example image of text being entered after interacting with a resource within the virtual world. In the left image, an office PC has been clicked (dashed rectangle), and the user is now recording the actions they perform with that PC (dashed rectangle at bottom left). In a similar manner in the right image, a lecturer has been clicked.

Despite this free form entry method, the text input is constrained via interactions with relevant human or non-human resources. The direct manipulation interface provides a natural, mnemonic approach to interact directly with the objects having most affordance for the activity [7,24]. A further benefit of the usage of direct manipulation interfaces – also called WYSIWYG interface (what you see is what you get) – is that the objects are visible and hence the interviewees do not need to remember complex syntax [24,12]. This way, novices can also learn the functionalities quickly. For virtual worlds, the direct manipulation principles are very helpful in providing the feeling of direct involvement with the simulation [24,23,12]. Such involvement, we believe, will result in a more consistently defined set of activities, due to the priming interaction with a visually familiar representation of resources.

The interaction approach is implemented as a single script, programmed in the Open Simulator scripting language. It is able to invoke PHP service calls within a WAMP system. In each case the information recorded is drawn from the virtual world data for the object. A single script is used for each object, which packages the name and read text from the user, saving it on the WAMP server, resulting in an easy interaction configuration of the world. An object to be used, for example the PC shown in Figure 3, has this single script attached to it via a menu interface. From then on, it is able to provide process elicitation

information as text descriptions typed in by the user. This can be done for every resource in the world that is relevant to the process being elicited.

It should be noted here, for clarity, that the virtual world does not contain animated functionality, it simply represents the visual state of the world, with mostly static artefacts. None of the interactions change the state of the world, they are simply user interface functions for drawing out process activity information from the interviewees, and so only represent the world in an as-is state. Furthermore, the intention is that the interactions in this prototype are to be used in an interviewing scenario where the person is being interviewed to describe the process from their perspective. While human and software services have representations within the virtual world, no human resources are animated. Thus agent simulation and modelling capabilities regarding to-be components are out of the scope of this present research and have been left for future work by the researchers.

Continuing on, each activity is specified in turn by interacting with (clicking on) the resources and typing to them the description of the activity. This activity description (predicate), along with the resource description, is stored in a database. Each table entry will encode the action taken with the resource, and any business objects used in the process for an initial data perspective. In addition, the stored text string provides a simple sentence structure which can be utilised later in natural language processing algorithms, as shown below.

<Subject>, <Object>, <Predicate>

The subject is the user of the system (automatically recorded), the object is the human, non-human resource utilised in the interaction (automatically recorded), leaving a free form predicate string to describe what the subject is doing with the object in question. A detailed description of the string, as placed in the database, is given later in Section 3.2. An example from our case study is:

*<Jan>, <Office PC>, <Send application for PC software installation>,
<Doc:>"IT Services Software Installation Form">*

As mentioned previously, one of the major challenges in process elicitation is the garnering of consistent data [13,3]. The need for consistent nomenclature is assisted here by the selection of named resources in the environment and automatic recording of the subject's name; any activity is constrained to be related to modelled human and non-human resources within the organisation. However, this approach is balanced by a need for free form expression, so that interviewees are able to provide a subjective view of their to-do list that may be outside of a specified action, or may contain other useful contextual metadata.

3.2 World Context Information Data Modelling

Once these strings have been captured, they must be stored in a manner that will facilitate future merging and processing. This stored information, gathered

from traversing the virtual world, provides further information to the analyst regarding the context of the activities, in particular, their equivalence and temporal ordering. The intention is to allow multiple people to use the environment, record their traces and then use these results to generate process models. While context information has been modelled before, from the perspective of activity equivalence determination [21], we extend this model to include world context information that will enhance the identification of similar activities for merging. Each of the to-do activities recorded will contain the following information:

- **Subject:** the name of person carrying out the elicitation activity, drawn automatically from configuring the avatar representation in the environment.
- **Object:** the name of the resource being interacted with in the environment; for example, printer, table, x-ray machine, reception desk, and so on.
- **Predicate:** a free form string used to describe the activity performed with the object in the activity, for example, print class attendance sheet.
- **Data Object:** the business object used, representing a data perspective within the activity, for example, enrolment application form.

In addition to the information listed above for the to-do strings, the following is stored in the database:

- **Position:** the x,y,z coordinate of the event recorded by the subject in the virtual world, measured in metres.
- **Timestamp:** time and date of the recorded event during the interview, providing a relative temporal order to activities.

These elements have been implemented in WAMP as an SQL database table. Note that the Data Object component is assumed to be within the predicate string in this implementation. The table can be written out as an .xes format event log if required.

4 World Context and Similarity Measures

Once the world has been modelled and used within the interview sessions, the data extracted needs to then be processed effectively to provide useful log data for later analysis. Previous work has defined equivalence as a context analysis with respect to the semantics of the underlying services being executed by an activity [21]. We take a similar approach to answer R3 (see Introduction), but formalise the context operators from the perspective of the extra world context information previously listed. This comparison process is supported in part by the nature of the interface to the virtual world, as it constrains the specification of activities to key resources and locations, making the process of analysis for determining merge points a lot easier. The interface, as a by product, also assists with the challenges of precision and reproducible log data, by constraining the user somewhat, to a more regular expression of key activities within the process, in itself facilitating the equivalence formalism.

Definition 1 (World context). Let \mathcal{A} be the set of all activities executable within a world domain of activities. An activity $A \in \mathcal{A}$ is defined as tuple $A := (S, O, P, D, dT, W, T)$, drawn from the information typed by users of the virtual world during an interview session, where:

- S - subject, name of agent carrying out the elicitation activity.
- O - object, the name of the resource being interacted with in the environment to execute the activity.
- P - predicate, a verb used to describe the activity performed with the object in the activity.
- D - set of data objects, representing a data perspective within the activity (set of input/output data related to activities).
- $dT : D \mapsto \{\text{input, output}\}$ - function dT maps each data object in D onto its type, i.e., input or output data
- $W \subseteq \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ - world position, three-dimensional coordinates of the activity recorded.
- T - timestamp of the activity.

For this paper we focus on similarity as a measure, due to the need to specify, as best as possible, inexact comparison capabilities for the context metadata, in particular, locations in space, and references to labeled objects; we therefore use a small set of measures directly from [26].

Definition 2 (Label similarity). For activity A tuple elements S, P, D - O and W are defined uniquely below in Definitions 4 and 5. We use a modified version of the string edit measure definition, from [26], where l_1, l_2 are two labels, and let $|l|$ represent the number of characters in the label l . The string edit distance of the labels is denoted $ed(l_1, l_2)$ and is the minimal number of atomic string operations needed to transform l_1 into l_2 or vice versa. The atomic string operations are: inserting a character, deleting a character or substituting a character for another. The label feature similarity of l_1 and l_2 , denoted $lsim(l_1, l_2)$ is:

$$lsim(l_1, l_2) = 1 - \frac{ed(l_1, l_2)}{\max(|l_1|, |l_2|)}$$

As the context data provided in a virtual world is spatial in nature, there is a need for a similarity operator that takes into account the locations of activities and resources within the environments, so a positional similarity measure must be developed to assess the relative locations of entities within the virtual world.

Definition 3 (Euclidean distance feature similarity). Let W_{a1}, W_{a2} be the world positions of two activities $a1, a2 \in \mathcal{A}$. Then the distance $edist$ between W_{a1}, W_{a2} can be determined by

$$edist(W_{a1}, W_{a2}) = |W_{a1} - W_{a2}|$$

where $|W_{a1} - W_{a2}|$ represents the euclidean distance between the two 3D points.

When world object similarities are to be considered, there is the need to combine label and spatial similarity estimates in order to have an accurate estimation of the similarity of two objects in the virtual world. The strong assumption is that objects, with the same label and same location, are the same object. Objects with similar labels, but in a different location, are different objects, being used for a similar activity. These two similarity estimates are combined into a ratio, as shown in the following, allowing the distance between the log entries, measured as $edist$, to suppress any label similarity effect.

Definition 4 (Object feature similarity). *Let O_{a1}, O_{a2} be the objects associated with two activities $a1, a2 \in \mathcal{A}$ and W_{a1}, W_{a2} be the associated world positions. Then object similarity $osim$ between $a1, a2$ can be defined as follows:*

$$osim(a1, a2) = \frac{lsm(label(O_{a1}), label(O_{a2}))}{edist(W_{a1}, W_{a2}) + 1}$$

where $label(O)$ returns the label of object O .

The interpretation of $osim$ is that if the object labels and world positions are similar, then $osim$ returns a high similarity. However, if the object labels are similar, but the world positions are not, then object similarity is decreased. This reflects, for example, a case of similar devices being on different floors. In case that lsm of two object labels is 1 and the distance between their world positions is 0, $osim$ yields a similarity of 1. In case the label similarity is 0, $osim$ becomes 0. For $lsm = 0.5$ and $edist = 0.5$, $osim = 0.33$.

5 Formalization and Analysis of Hints within the Logs

We now initially validate the technical method via examples and descriptions of functionality [11] as an initial presentation of the capabilities of the new BP-MEVW approach. A number of detailed examples of using the virtual world are used to generate augmented to-do lists for later merging into process models. For this initial research, we have a two fold validation process. Firstly, we map the hints listed in Section 2 to a list linking the metadata features with related similarity operations. This forms an initial set of comparison operators that can be applied to the metadata to generate hints. Secondly, we provide examples of these hints being generated for activity specifications drawn from the two example use cases. Analysis of examples for the Vienna use case [13] are drawn from a codebook generated to fully define process activities. The codebook contains key terms processed by the interviewer, in order to provide a basis for activity merging and alignment. Raw logs provided by the BPI2011 challenge, as previously mentioned, were used as a source of activity specifications and resource information for the Dutch healthcare case study. Log examples have been selected from this health example, to best highlight issues around the *tools* and *reoccurring activity* hints.

The following sections detail the hints, showing the similarity operator developed and their application to data from either use cases. In each case, the examples refer to unary or binary operations performed on two activities $a1$ and

$a2$ from the set of \mathcal{A} . The examples have each been labeled CS for Vienna Computer Science and DAH for Dutch Academic Hospital respectively. The bolded example data for CS or DAH are the extra context metadata extracted from the virtual world, over and above any interview information that would be gained using a normal BPME approach.

5.1 Data Transfer Operator

Let $a1, a2 \in \mathcal{A}$ be two activities, S_{a1}, S_{a2} be the subjects carrying out $a1, a2$, and D_{a1}, D_{a2} the data objects of $a1, a2$ (cf. Definition 1). Then a data transfer between $a1$ and $a2$ can be detected from the log based on Equation (1) using label similarity metrics lsm defined in Definition 2 and threshold $lcutoff$:

$$lsm(D_{a1}, D_{a2}) > lcutoff \wedge S_{a1} \neq S_{a2} \quad (1)$$

CS Example : P_{a1} = “Mail lecturing contract to lecturer,” P_{a2} = “Receipt of the lecturing contract in my post office box” S_{a1} = ”**AdministrativeCoordinator**,” S_{a2} = ”**Lecturer**,” D_{a1} and D_{a2} = ”**lecturing contract**”

Obviously, the two subjects are different and the label similarity between the data elements D_{a1} and D_{a2} is 100%. This indicates that a data perspective element D has been shared between two different subjects. In addition, the use of the interactive virtual world interface (see Figure 3) ensures the subjects chosen are predefined, and indeed different, providing a clearer indication of such data transfer events upon log analysis.

5.2 Group Tasks

Let P_{a1}, P_{a2} be two activity predicates written into the log of interest. Let further S_{a1}, S_{a2} be the subjects and W_{a1}, W_{a2} be the world positions associated with $a1, a2$ (cf. Definition 1). Then it can be detected from the log that (similar) activities $a1$ and $a2$ belong to a group task based on Equation (2) using Euclidean distance feature similarity metrics $edist$ defined in Definition 3 and thresholds $lcutoff, ecutoff$:

$$lsm(P_{a1}, P_{a2}) > lcutoff \wedge edist(W_{a1}, W_{a2}) < ecutoff \wedge S_{a1} \neq S_{a2} \quad (2)$$

CS Example : P_{a1} = “Participation in coordination meeting,” P_{a2} = “Coordination meeting” S_{a1} = ”**TeachingCoordinator**,” S_{a2} = ”**Lecturer**” $W_{a1} = <134, 131, 23>$, $W_{a2} = <134, 131, 23>$.

The distance between the predicates P of $a1$ and $a2$ turns out as 0.48 (i.e., $1 - 18/37$). The spatial distance between W_{a1}, W_{a2} is 0. The multiple different subjects in a shared 3D location for a similar activity hints at a group activity. As a consequence, activities $a1$ and $a2$ might be aggregated in the resulting model.

5.3 Delegation

Let P_a be an activity predicate contained in the log of interest and O_a be the object associated with a . Then a hint to a delegation can be found based on the following Rule (4):

$$\text{label}(O_a) \in H_{\text{Role}} \vee \text{label}(O_a) \in H_{\text{ID}} \quad (3)$$

where, H_{Role} is the set of human resource roles allocated to the running case (e.g., "Lecturer"), H_{ID} is the system ID of a human resource (e.g., "Lect2314") and $\text{label}(O)$ returns the label of object O as a string.

CS Example : P_{a1} = "A lecturer of our lecturer team books a lecture hall for all of us" O_{a1} = "Lecturer".

As the object O in the interaction event recorded is a human role found in H_{Role} , then this is a strong hint of a delegation. In addition, we note the use of the virtual world tends to constrain the interviewee to identified roles.

5.4 Tools

Let P_{a1}, P_{a2} be two activity predicates written into the log of interest. Further, let O_{a1}, O_{a2} be the objects associated with $a1, a2$. Then it can be detected that the two activities $a1, a2$ use the same tools based on the following Equation (4) using label similarity and object similarity with thresholds $lcutoff$ and $ocutoff$:

$$lsim(P_{a1}, P_{a2}) > lcutoff \wedge osim(O_{a1}, O_{a2}) > ocutoff \quad (4)$$

DAH Example : P_{a1} = "ultrasound scan abdomen," P_{a2} = "ultrasound hip," O_{a1} = "UltrasoundMachine," O_{a2} = "UltrasoundMachine." $W_{a1} = <120, 131, 10>$, $W_{a2} = <120, 131, 10>$.

The similarities for the above example turn out as follows: $lsim = 0.26$ (ie. $1.0 - 17/23$), $osim = 1$ (i.e.. identical in name and location). Resource alignment in an interactive virtual world hospital will be better defined and aligned with activities due to direct interactions with precisely named resources.

5.5 Reoccurring Activities

Let P_{a1}, P_{a2} be two activity predicates written into the log of interest. Further, let S_{a1}, S_{a2} be the subject, O_{a1}, O_{a2} be the objects, and D_{a1}, D_{a2} be the data objects associated to $a1, a2$. Then it can be detected that P_{a1}, P_{a2} belong to a reoccurring activity based on the following Equation (5) using label similarity and object similarity with thresholds $lcutoff1$, $lcutoff2$, and $ocutoff$:

$$\begin{aligned} lsim(P_{a1}, P_{a2}) &> lcutoff1 \wedge S_{a1} = S_{a2} \\ &\wedge osim(O_{a1}, O_{a2}) > scutoff \wedge lsim(\text{label}(D_{a1}), \text{label}(D_{a2})) > lcutoff2 \end{aligned} \quad (5)$$

where $\text{label}(D)$ yields the label of data object D.

This definition corresponds to the attribute equivalence proposed in [22], and is detailed for logs in this approach.

DAH Example : P_{a1} = "Administrative Fee," P_{a2} = "Administrative Fee," O_{a1} = "Reception," O_{a2} = "Reception," S_{a1} = "Patient," S_{a2} = "Patient," D_{a1} = "Receipt." D_{a2} = "Receipt." $W_{a1} = \langle 113, 131, 0 \rangle$, $W_{a2} = \langle 113, 131, 0 \rangle$.

Label similarity between the predicates of the activities and the labels of the associated data elements turns out as 1. The same holds for the object similarity, as it is the reception desk at the same location. The associated subjects are the same. Hence, it can be concluded that the activity predicates in the log really belong to the same activity that has been executed in a repetitive way.

The equivalent context information in this administrative fee charging medical case (subjects, resources and data objects) may highlight loops due to contextual similarities.

5.6 Time Notes

The hint "Time Notes" is illustrated based on the following example.

CS Example : P_{a1} = "inquiry for technical requirements," P_{a2} = "apply for licenses" $T_{a1} = \text{"28/03/2014 5:20 pm,"}$ $T_{a2} = \text{"28/03/2014 5:22 pm."}$

Our conjecture, to be confirmed by empirical experimentation, is that due to the visual priming effect of an interactive virtual world, and automatic time stamps, $a1$ should be correctly positioned before $a2$, reducing previously mentioned issues with temporal ordering.

5.7 Activity Merge

Let P_{a1}, P_{a2} be two activity predicates written into the log of interest. Further, let O_{a1}, O_{a2} be the objects, and W_{a1}, W_{a2} be the world positions associated with $a1, a2$. Then it can be detected that $a1, a2$ might be candidates for an activity merge based on the following Equation (6) using label and object similarity as well as Euclidean distance with thresholds $lcutoff$, $ocutoff$, and $ecutoff$:

$$\begin{aligned} lsim(P_{a1}, P_{a2}) &> lcutoff \wedge osim(O_{a1}, O_{a2}) > ocutoff \\ &\wedge edist(W_{a1}, W_{a2}) < ecutoff \end{aligned} \quad (6)$$

Note that the distance between the world positions is implicitly considered in $osim$. However, as illustrated by the following example, it is important to emphasize that the work associated with the activities was performed in the same location.

CS Example : P_{a1} = "course design," P_{a2} = "describe course" O_{a1} = "Home PC," O_{a2} = "Tablet PC" $W_{a1} = \langle 90, 200, 23 \rangle$, $W_{a2} = \langle 90, 200, 23 \rangle$.

The similarities for the above example turn out as follows: $lsim = 0.4$ (ie. $1.0 - 9/15$), $osim = 0.4$ (ie. $(1.0 - 6/10)(0.0 + 1.0)$). If the subjects are found to be doing this work in the same location, with similarly labelled services or resources, then this is a hint that the activities may be the same.

6 Related Work

Process model elicitation is an important stage in the BPM-life cycle, and its contribution to successful modelling exercises cannot be underestimated. However, we note that not many software tools have been developed beyond descriptive methodologies to elicit this information [13], with even experiential papers as guidelines [1]. There has been some work to use forms of simulation and recommendation to extract information from users [17], but by and large, elicitation methods are usually implemented as workshops with stakeholders, or interviews, one-on-one [13].

From an HCI perspective, some tools have been developed to engage users in process elicitation, including collaborative tables and tangible modelling approaches. A stakeholder driven collaborative modelling table approach to process elicitation, that seeks to allow stakeholders to model their processes using a subjective form of grammar, has been developed [25]. The table seeks to facilitate discussion via tangible blocks mounted on an interactive light table, that assists in specifying processes from a subjective point of view. In related research, tangible modelling approaches have been developed by Luebbe and Weske [14], whereby plastic tiles marked with pens are used as modelling artefacts. While these tangible techniques have been found to engage users, they do not progress past variations on standard modelling grammars, and do not incorporate any novel 3D interfaces beyond those based on graph like process grammar representations.

Previous work has been carried out by the authors into using virtual worlds for various aspects of process modelling, including modelling and remote collaboration [20], and simulation [9] but they have not explored the priming aspects of virtual worlds in process elicitation, except in a high level theoretical manner [10]. The research described in this paper is a first attempt at operationalising such an approach with a formalism and case study examples to illustrate the potential of such an approach.

7 Conclusion/Future Work

We have motivated in this paper the need for better approaches to expert knowledge elicitation, as it stands with process modelling tasks. Addressing R1, we have theoretically shown the potential efficacy of virtual worlds as tools for tacit knowledge elicitation as modelling hints. This tacit knowledge extraction is expected to improve via the immersion of the stakeholder into a view that promotes recall of subtle information in their world, viz. a virtual world of their work. Addressing R2, from our analysis of challenges in log processing, we have developed an approach to using a virtual world model of a workplace to elicit more consistent and accurate information from an interviewee. This approach has been encoded as a formalism for processing via virtual world context metadata, which was then mapped to explicit hint mechanisms for use by business analysts in their process model merging and alignment tasks. Addressing R3, examples

were drawn from CS and DAH logs, showing the potential for this approach to be integrated into an automated software system to advise analysts.

Potential limitations to the validity of this preliminary work arise from a lack of testing with a cohort of users to determine the effects of such a virtual world approach on process elicitation, including issues around usability and a detectable improvement in models elicited. Therefore, validation with a user cohort will be the next major aim of this work, in particular, to gauge the ability of this approach to improve process activity recall, and to improve issues with activity specification ambiguities, temporal granularity and accuracy in activity ordering. Furthermore, future work involving the integration of online lexicons (eg. RadLex - radlex.org) for user input text validation, may prove effective in assisting with quality control of resource and activity specifications. We expect that a future implementation may apply this context data to a process mining algorithm (such as the fuzzy miner [8]) to provide merge possibilities from mined logs.

References

1. Becker-Kornstaedt, U.: Towards systematic knowledge elicitation for descriptive software process modeling. In: Bomarius, F., Komi-Sirviö, S. (eds.) PROFES 2001. LNCS, vol. 2188, pp. 312–325. Springer, Heidelberg (2001)
2. Binder, M., Dorda, W., Duftschmid, G., Dunkl, R., Fröschl, K.A., Gall, W., Grossmann, W., Harmankaya, K., Hronsky, M., Rinderle-Ma, S., Rinner, C., Weber, S.: On analyzing process compliance in skin cancer treatment: An experience report from the evidence-based medical compliance cluster (EBMC²). In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 398–413. Springer, Heidelberg (2012)
3. Bose, R., Mans, R., van der Aalst, W.: Wanna improve process mining results? In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 127–134 (April 2013)
4. Burdea, G., Coiffet, P.: Virtual Reality. Wiley, New York (2003)
5. Clark, A.: Supersizing the Mind: Embodiment, Action and Cognitive Extension. Oxford University Press (2010)
6. Dunkl, R.: Data improvement to enable process mining on integrated non-log data sources. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST. LNCS, vol. 8111, pp. 491–498. Springer, Heidelberg (2013)
7. Galitz, W.O.: The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. John Wiley & Sons (2007)
8. 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)
9. Guo, H., Brown, R., Rasmussen, R.: Virtual worlds as a model-view approach to the communication of business processes models. In: CAiSE Forum, pp. 66–73 (2012)
10. Guo, H., Brown, R., Rasmussen, R.: A theoretical basis for using virtual worlds as a personalised process visualisation approach. In: Franch, X., Soffer, P. (eds.) CAiSE Workshops 2013. LNBIP, vol. 148, pp. 229–240. Springer, Heidelberg (2013)
11. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly 28(1), 75–105 (2004)

12. Hutchins, E.L., Hollan, J.D., Norman, D.A.: Direct manipulation interfaces. *Hum.-Comput. Interact.* 1(4), 311–338 (1985)
13. Kabicher, S., Rinderle-Ma, S.: Human-centered process engineering based on content analysis and process view aggregation. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 467–481. Springer, Heidelberg (2011)
14. Luebbe, A., Weske, M.: Tangible media in process modeling – A controlled experiment. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 283–298. Springer, Heidelberg (2011)
15. Luebke, D., Watson, B., Cohen, J.D., Reddy, M., Varshney, A.: Level of Detail for 3D Graphics. Elsevier Science Inc. (2002)
16. Mans, R.S., Schonenberg, M.H., Song, M., van der Aalst, W.M.P., Bakker, P.J.M.: Application of process mining in healthcare a case study in a dutch hospital. In: Fred, A., Filipe, J., Gamboa, H. (eds.) BIOSTEC 2008. CCIS, vol. 25, pp. 425–438. Springer, Heidelberg (2009)
17. Martinho, D., Silva, A.R.: A recommendation algorithm to capture end-users' tacit knowledge. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 216–222. Springer, Heidelberg (2012)
18. Miller, M.: The magical number seven, plus or minus two. *Psychological Review* 63, 81–97 (1956)
19. Moody, D.: The “physics” of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng.* 35(6), 756–779 (2009)
20. Poppe, E., Brown, R., Recker, J., Johnson, D.: Improving remote collaborative process modelling using embodiment in 3d virtual environments. In: Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling, APCCM 2013, vol. 143, pp. 51–60. Australian Computer Society, Inc., Darlinghurst (2013)
21. Rinderle-Ma, S., Reichert, M., Jurisch, M.: Equivalence of web services in process-aware service compositions. In: IEEE International Conference on Web Services, ICWS 2009., pp. 501–508 (July 2009)
22. Rinderle-Ma, S., Reichert, M., Jurisch, M.: On utilizing web service equivalence for supporting the composition life cycle. *International Journal of Web Services Research (IJWSR)* 8(1), 41–67 (2011)
23. Shneiderman, B., Plaisant, C.: Designing the User Interface: Strategies for Effective Human-Computer Interaction, 4th edn. Addison-Wesley Longman (1998)
24. Stone, D., Jarrett, C., Woodroffe, M., Minocha, S.: User Interface Design and Evaluation. Morgan Kaufmann (2005)
25. Wachholder, D., Oppl, S.: Stakeholder-driven collaborative modeling of subject-oriented business processes. In: Stary, C. (ed.) S-BPM ONE 2012. LNBP, vol. 104, pp. 145–162. Springer, Heidelberg (2012)
26. Yan, Z., Dijkman, R., Grefen, P.: Fast business process similarity search. *Distributed and Parallel Databases* 30(2), 105–144 (2012)

Deployment of Service-Based Processes in the Cloud Using Petri Net Decomposition

Sami Yangui¹, Kais Klai², and Samir Tata¹

¹ Institut Mines-Telecom. Telecom SudParis. UMR CNRS Samovar. Evry, France

{Sami.Yangui,Samir.Tata}@telecom-sudparis.eu

² LIPN. Galilee Institute. University of Paris 13. UMR CNRS 7030. Villetaneuse, France

kais.klai@lipn.univ-paris13.fr

Abstract. Cloud Computing is a new distributed computing paradigm that consist in provisioning of infrastructure, software and platform resources as services. Platform services are limited to proprietary or specific programming frameworks and APIs. This issue is not adequate for the deployment of service-based processes which are likely to be composed of a diverse and heterogeneous set of services. In this paper, we propose a new approach to provision appropriate platform resources in order to deploy service-based processes in existing Cloud platforms. Our approach consists in slicing a given process to deploy into a set of elementary services through a Petri net decomposition approach. Source codes of obtained services are generated. After that, the services are packaged in our already developed service micro-containers and deployed in any target PaaS. For the slicing, we defined algorithms to slice their correspondent Petri net into a set of dependent WF-nets and to determine the orchestration to follow for their execution. We also provided the proof of preservation of initial business process semantics when executing the WF-nets. To illustrate and show the feasibility of our proposition, we provide a realistic use case scenario, i.e. Shop process deployment in Cloud Foundry PaaS.

Keywords: Service deployment, Business processes, Cloud Computing, Petri Net.

1 Introduction

Cloud Computing is a new distributed computing paradigm. It differs from traditional ones on the fact that (1) it is massively elastic, (2) it can be encapsulated as an abstract entity that delivers different levels of services to customers outside the Cloud, (3) it is driven by economies of scale and (4) it can be dynamically configured (via virtualization or other approaches) and delivered on demand [2]. Cloud services can be of infrastructure type (aka Infrastructure as a Service or IaaS), of platform type (aka Platform as a Service or PaaS) or of software type (aka Software as a Service or SaaS). PaaS is a sort of a re-usable framework which provides one or more platform components as a service. The associated service delivery model allows Cloud end users to rent virtualized platform resources and services for developing and/or running applications.

In this work, we focus on the provisioning and the deployment of platform resources mainly for the deployment and running of service-based processes. Such applications

are based on Service-Oriented Architecture (SOA) [4] and consist of assembling a set of elementary services using appropriate service composition specifications like Web Services Business Process Execution Language (BPEL) [5] or Business Process Model and Notation(BPMN) [6].

However, all Cloud platforms we have studied, e.g. Cloud Foundry, OpenShift, Windows Azure, Google App Engine, Heroku, Jelastic, etc., suffer from limitations and restrictions to support deployment and running of such service-based processes. On one side, these Cloud platforms are limited to proprietary or specific programming frameworks and APIs. On the other side, service-based processes are likely to be composed of a diverse and heterogeneous set of services. Therefore, the use of those platforms is difficult since Cloud end users need to use the related programming frameworks and APIs before using the Cloud to provision and deploy such applications. This makes the deployment of already developed services difficult if not impossible. In addition, enforcing cloud properties such as elasticity or multi-tenancy needs to develop new versions of existing application servers and process engines or to develop new ones [1]. Indeed, we think that it is not necessary to make a service-based process elastic or multi-tenant while the need can be to get a single/specific service elastic or multi-tenant. We have shown in previous works that it is more interesting in terms of performance (running time and memory consumption) to handle elasticity [23] and autonomic management [24,22] at the service level rather than at the service-based application level.

While we have proposed a process to split, deploy and execute service-based applications described in Service Component Architecture [9], our objective in this paper is to propose a novel approach to split, deploy and execute service-based applications described as business processes, e.g. described in Business Process Execution Language (BPEL) [5] or Business Process Model and Notation (BPMN) [6]. Broadly speaking, our defined approach consists of (1) transforming a given service-based process in an equivalent Petri Net, (2) slicing the Petri Net into a set of sub nets according to a slicing algorithm that we have defined. Each sub net represents an elementary service of the initial process, (3) generating the source code corresponding to each sliced service, (4) packaging the obtained services into service micro-containers [11] [10] and finally, (5) deploying the generated micro-containers in the target PaaS through COAPS API [12].

The remainder of the paper is organized as follows: In Section 2, we present some preliminary notions on Petri nets. In Section 3, we present and detail the different steps of our defined approach to deploy service-based processes in Cloud platforms. Each one of these steps is illustrated thanks to our motivating example. In Section 4, we discuss related work and highlight classical PaaS provider limitations regarding resources provisioning for service-based processes. Finally, in Section 5, we conclude the paper and we give directions for future work.

2 Preliminaries

In our proposed deployment process, one of our operations we perform on service-based processes is slicing (as it is described in Section 3). Consequently, we propose to use Petri nets to formally describe service-based processes. This allows us to preserve semantics of sliced processes. Since several modeling languages map to Petri nets (for

BPEL, see e.g. [28], for BPMN, see e.g. [32]), our approach is relevant for a very broad class of modeling languages or service-based processes. In this following, we present some preliminary notions on Petri nets.

2.1 Petri Nets

Definition 1. A Petri net (Place-Transition net) is a bipartite directed graph $N = \langle P, T, F, W \rangle$ where:

- P is a finite set of places (circles) and T a finite set of transitions (squares) with $(P \cup T) \neq \emptyset$ and $P \cap T = \emptyset$,
- A flow relation $F \subseteq (P \times T) \cup (T \times P)$,
- $W : F \rightarrow \mathbb{N}^+$ is a mapping that assigns a positive weight to any arc.

Each node $x \in P \cup T$ of the net has a pre-set and a post-set defined respectively as follows: $\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$, and $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$. Adjacent nodes are then denoted by $\bullet x^\bullet = \bullet x \cup x^\bullet$. Given a set of nodes S , $|S|$ denotes the cardinality of S i.e. (the number of elements belonging to S).

The incidence matrix C associated with the net is defined as follows : $\forall (p, t) \in P \times T : C(p, t) = W(t, p) - W(p, t)$. A marking of a Petri net N is a function $m : P \rightarrow \mathbb{N}$. The initial marking of N is denoted by m_0 . The pair $\langle N, m_0 \rangle$ is called a Petri net system. A transition t is said to be enabled by a marking m (denoted by $m \xrightarrow{t}$) iff $\forall p \in \bullet t, W(p, t) \leq m(p)$. If a transition t is enabled by a marking m , then its firing leads to a new marking m' (denoted by $m \xrightarrow{t} m'$) s.t. $\forall p \in P : m'(p) = m(p) + C(p, t)$. The set of markings reachable from a marking m in N is denoted by $R(N, m)$. A run of marked petri net $\langle N, m_0 \rangle$ is a sequence $\sigma = t_1 \dots t_n$ such that $m_0 \xrightarrow{t_1} m_1 \dots \xrightarrow{t_n} m_n$. The language of a marked net $\langle N, m_0 \rangle$ is the set of its runs and is denoted by $L(\langle N, m_0 \rangle)$.

Two Petri nets $N_1 = \langle P_1, T_1, F_1, W_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2, W_2 \rangle$, sharing a subset of transitions (resp. places), can be composed by merging these transitions (resp. places) leading to a Petri net regrouping the local attributes of N_1 and N_2 . In the approach presented in this paper, we manage to compose Petri nets with disjoint sets of transitions (i.e., sharing only some places).

Definition 2. Let $N_1 = \langle P_1, T_1, F_1, W_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2, W_2 \rangle$ be two Petri nets such that $P_1 \cap P_2 \neq \emptyset$ and $T_1 \cap T_2 = \emptyset$. The composition of N_1 and N_2 by merging of common places, denoted by $N_1 \oplus N_2$, is a Petri net $\langle P, T, F, W \rangle$ where $P = P_1 \cup P_2$, $T = T_1 \cup T_2$, $F = F_1 \cup F_2$ and $W : F_1 \cup F_2 \rightarrow \mathbb{N}^+$ is the mapping assigning the weight $W_i(f)$ to any arc in F_i , for $i \in \{1, 2\}$.

The decomposition of a given Petri net into two (or more) subnets corresponds to the dual operation. Again, the two main decomposition approaches are based on the splitting of a Petri net into two (or more) subnets that share a subset of places or/and a subset of transitions. The sharing of transitions represents synchronization (rendezvous) between two components while the sharing of places (buffers) represents an asynchronous communication between components.

2.2 WF-Nets

We use a particular Petri net for modeling the control-flow dimension of service-based processes. This is a variant of (*WF-nets*) that have been introduced in [33].

Definition 3 (Wf-net). Let $N = \langle P, T, F, W \rangle$ be a Petri net. N is said to be a workflow net (Wf-net) if

- there is a set of source place $I \in P$ s.t. $\bullet I = \emptyset$ and a set of sink places $O \in P$ s.t. $O^\bullet = \emptyset$, and
- for any node $x \in P \cup T$, for any source place $i \in I$ and for any sink place $o \in O$, there exists a path from i to o which passes through x .

Note that the sole difference between the WF-nets introduced in [33] and the above definition is the fact that, we allow a WF-net to have several source places and/or several sink places. This has no effect on the semantics of the obtained model but is more convenient for our decomposition approach. As a special kind of Petri nets, WF-nets have the same semantics described above. Its behavior can then be represented by its reachability graph. As far as the behavior of a workflow is concerned, the corresponding Wf-net is associated to an initial marking where only the source places are marked with a single token. Besides, a *final marking* of Wf-net is every one, that is reachable from a such initial marking, where each sink place is marked and none of the other places is.

3 Service-Based Processes Deployment Approach

In this section, we detail our defined approach to deploy and execute service-based processes in Cloud platforms. This approach consists of five steps:

1. Transforming of the process to deploy into an equivalent Petri Net,
2. Slicing the Petri Net into a set of autonomous and communicating sub nets,
3. Generating of the source code of each one of the obtained services,
4. Packaging of codes of the resulted services into service micro-containers,
5. Deploying of the micro-containers on a target Cloud platform.

In the following, we present a running example used to illustrate these steps.

3.1 Running Example

To illustrate our contribution, we consider an online shop process as example. The correspondent BPEL description of this process is presented in Fig. 1.

When the online shop receives information from a customer, business strategy of the process distinguishes between already known customers and new customers. In case of a known customer, an appropriate branch of the process is executed (lines 9-14): first, the shop receives an order (line 11), and then it sends the invoice to the customer (line 12). In case of a new customer (lines 15-26) the shop initiates two tasks concurrently: in the first task (Sequence 1, lines 17-20) the shop first receives the order (line 18) and then confirms it (line 19). In the second task (Sequence 2, lines 21-29) the shop receives the terms of payment (line 22) before it sends an invoice to the customer (line 23). In either case the shop finally sends the delivery information to the customer (line 28).

```

1 <process name="Online Shop">
2   <partnerLinks>
3     <partnerLink name="client"/>
4   </partnerLinks>
5   ...
6   <sequence>
7     <receive portType="onlineshop" operation="login">
8     <switch>
9       <case condition="known customer">
10        <sequence>
11          <receive partnerLink="client" operation="order"/>
12          <invoke partnerLink="client" operation="invoice"/>
13        </sequence>
14      </case>
15      <otherwise>
16        <flow>
17          <sequence>
18            <receive partnerLink="client" operation="order"/>
19            <invoke partnerLink="client" operation="confirm"/>
20          </sequence>
21          <sequence>
22            <receive partnerLink="client" operation="terms"/>
23            <invoke partnerLink="client" operation="invoice"/>
24          </sequence>
25        </flow>
26      </otherwise>
27    </switch>
28    <invoke partnerLink="client" operation="deliver"/>
29  </sequence>
30 </process>

```

Fig. 1. Shop BPEL process descriptor

3.2 Generating the Petri Net

This step consists in generating a Petri Net from a given service-based process (BPEL process in our running example). To perform the BPEL to Petri Net transformation, we use the BPEL2PN¹ tool. BPEL2PN tool is a Java-based compiler that transforms a process specified in BPEL into a Petri net according to the Petri net semantics [25] [26]. The output format of BPEL2PN is a Petri net in the data format of the Petri net-based model checker LoLA². LoLA also offers the opportunity to write out the Petri net into the standard interchange format for Petri nets, the Petri Net Markup Language (PNML) [29] [30].

¹ <http://www2.informatik.hu-berlin.de/top/bpel2pn/index.html>

² <http://www2.informatik.hu-berlin.de/top/lola/lola.html>

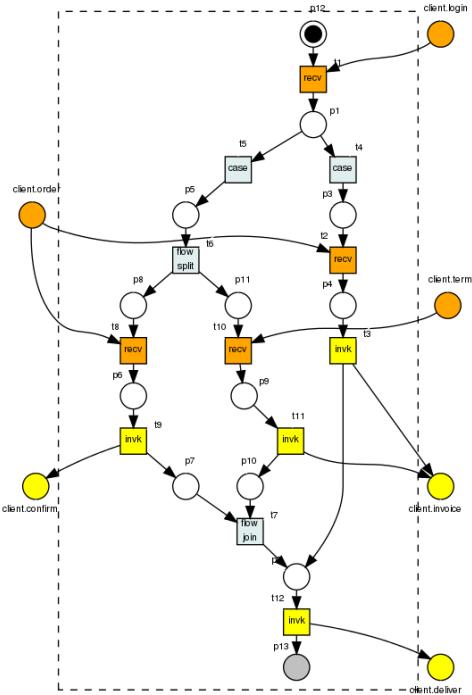


Fig. 2. The Petri Net corresponding to the shop process

The generated Petri net from our shop process is schematized in Fig. 2. Shop process activities are represented in the Petri net as transitions and the call transfer between these activities are represented as places. The usefulness of the Petri Net transition is to ensure formalizing the interactions between all the business process activities while keeping intact its semantics.

3.3 Slicing of the Obtained Petri Net

The decomposition of a WF-net corresponding to a service-based process is based on an on-the-fly traversal of the whole Petri net, considered as a graph, and takes into account the following considerations:

1. each sub-net is a WF-net (according to Definition 3)
2. the decision to cut a current subnet and start the construction of a new one is based on the structure of the Petri net and on the nature of the process activities/services (respectively):
 - when the current node (place) has several output transitions i.e., it is a choice between two (or more) services/activities, (this may corresponds to an *IF..Else* structure in a process),

- when the current node (place) has several input transitions i.e. it is a conjunction of several exclusive activities/services,
 - when the current node (transition) has several output places i.e., it is the launch of two (or more) parallel services/activities, (this may corresponds to a *Flow* structure in a process),
 - when the current node (transition) has several input places i.e., it is a synchronization of several concurrent activities/services,
 - when the current node (transition) is the waiting point for the reception of some asynchronous message (this corresponds to a *Receive* activity/service),
 - when the current node (transition) is the synchronization point from some synchronous message, (this corresponds to the end of the *Flow* structure in the process),
3. given the resulting set of subnets, a dependency function, computed on-the-fly, allows to deduce the order of the execution of the services corresponding to these subnets. Thus, the combination of the set of subnets with such a function determines the semantics of the whole net.
 4. it is possible to compose back the obtained subnets, by merging shared places, to obtain the original whole WF-net. This allows to preserve the original behavior (semantics) of the Petri net and hence the corresponding service-based process.

Given a process model, which has been translated to WF-net, we use Algorithm 1 to slice the WF-net into several WF-subnets. The inputs of the algorithm are a current node (*curNode*), which can be either a place or a transition, a current subnet (*curServ*), to be decomposed, and the whole WF-net (*W*). The first call to Algorithm 1 is performed using the following input: the source place of the whole WF-net and a current WF-subnet which is empty. Algorithm 1 is recursive and uses the following functions: *NewService()* creates a new subnet and initialize its set of places, transitions and edges with the given parameters. The function *Dep(S, S')* determines the occurrence dependency between the subnets (services) S and S' . Such a dependency can be of three kinds: $S \rightarrow S'$ means that once the execution of S is finished, the execution of S' should start. $S \vee S'$ means that either S or S' is executed (exclusive or), and $S \parallel S'$ means that the execution of S and the execution of S' are concurrent (parallel). We also save the subnets generated by each node (*curNode*) in *BuiltSlice()*, so that the generation is done once for each encountered node (place or transition) (using the boolean function *Alreadytreated()*.

The algorithm is composed of three main phases. First, starting from the current node, the current subnet service (*curServ*) is incremented (by adding places and transitions) as long as we are following a linear (sequential) branch of the Petri net i.e. each encountered node has a single input, a single output, and is neither a receive transition nor a synchronous invoke activity (lines 3 – 6). When the first phase terminates, i.e. the current node has more than one input/output node. First, if it is a place then it is the output place of the current subnet (lines 7 – 9). Then, if this node has been already treated (lines 10 – 15), i.e. a decomposition is starting from this node or a predecessor node has already been performed, then, the set of built subnets must be executed before the current one, and are added as resulting from the decomposition of the initial node (*initNode*). Finally, two cases are considered depending on whether the current

Algorithm 1. Decomposition of a WF-net

Require: $initNode, curServ, W$

Ensure: a set of dependent WF-nets services

```

1: CutNode={receive, synchronous invoke}
2: curNode = initNode
3: while ( $|curNode^\bullet| = |^\bullet curNode| = 1$ ) and
   ( $curNode \notin CutNode$ ) do
4:   add { $curNode$ } to curServ
5:   curNode= $curNode^\bullet$ 
6: end while
7: if ( $curNode$  is a place) then
8:   add { $curNode$ } to curServ
9: end if
10: if Alreadytreated( $curNode$ ) then
11:   for all  $S \in BuiltSlices(curNode)$  do
12:      $Dep(curServ, S) \rightarrow$ 
13:     add  $S$  to  $BuiltSlices(initNode)$ 
14:   end for
15: end if
16: return
17:  $Alreadytreated(initNode) = true$ 
18: if ( $curNode$  is a place) then
19:   for all  $(t \in curNode^\bullet)$  do
20:     if NOT Alreadytreated( $t$ ) then
21:        $S = NewService(curNode, t)$ 
22:       Decomposition( $t, S, W$ )
23:     end if
24:   end for
25:   for all  $(t \in curNode^\bullet)$  do
26:     for all  $S \in BuiltSlices(t)$  do
27:        $Dep(curServ, S) \rightarrow$ 
28:       add  $S$  to  $BuiltSlices(initNode)$ 
29:       for all  $(t' \in curNode^\bullet \wedge t' \neq t)$  do
30:         for all  $S' \in BuiltSlices(t')$  do
31:            $Dep(S, S') = \vee$ 
32:         end for
33:       end for
34:     end for
35:   end for
36: end if
37: if ( $curNode$  is a transition) then
38:    $S = NewService(curNode, curNode, curNode^\bullet)$ 
39:    $Dep(curServ, S) \rightarrow$ 
40:   add  $S$  to  $BuiltSlices(initNode)$ 
41:   if ( $|curNode^\bullet| > 1$ ) then
42:     for all  $(p \in curNode^\bullet)$  do
43:       if NOT Alreadytreated( $p$ ) then
44:          $S' = NewService(p)$ 
45:         Decomposition( $p, S', W$ )
46:       end if
47:     end for
48:     for all  $(p \in curNode^\bullet)$  do
49:       for all  $S'' \in BuiltSlices(p)$  do
50:          $Dep(S, S'') \rightarrow$ 
51:         add  $S''$  to  $BuiltSlices(initNode)$ 
52:         for all  $(p' \in curNode^\bullet \wedge p' \neq p)$  do
53:           for all  $S''' \in BuiltSlices(p')$  do
54:              $Dep(S''', S'') = \parallel$ 
55:           end for
56:         end for
57:       end for
58:     end for
59:   else
60:     if NOT Alreadytreated( $curNode^\bullet$ ) then
61:       Decomposition( $curNode^\bullet, S, W$ )
62:     end if
63:     for all  $S' \in BuiltSlices(curNode^\bullet)$  do
64:       add  $S'$  to  $BuiltSlices(initNode)$ 
65:        $Dep(curServ, S') \rightarrow$ 
66:     end for
67:   end if
68: end if

```

node is a place (lines 18 – 36) or a transition (lines 37 – 68). Notice that, in the second case, we do not distinguish whether the current transition is a "normal" one or a receive/synchronous invoke transition.

Figure 3 illustrates our decomposition algorithm on the shop process while Table 1 gives the dependency function associated to subnets resulting from the decomposition.

Given the set of WF-subnets constructed by our algorithm and the computed dependencies between these subnets, one can orchestrate the execution of the whole process by executing these sub-processes separately. The main novelty in our approach is that such scheduling is not centralized. It is distributed on the different subnets in such a way that the currently executed sub-process is able to compute (using the dependency function) the set of sub-processes that must be enabled after it finishes its execution.

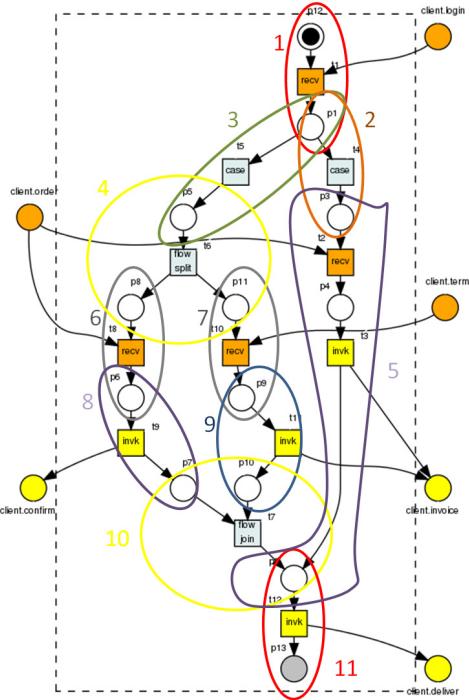


Fig. 3. The decomposed Petri Net corresponding to the shop process

Algorithm 2 accomplishes such a task. It has as inputs the current subnet to be executed, a set of subnets SN , and a dependency function Dep . Thus, an execution of the whole WF-net can be obtained by a call to Algorithm 2 with an empty subnet, the set of subnets and the dependency function issued from Algorithm 1. In this algorithm, *Choice* denotes any **maximal** subset of subnets, denoted by Ch , where any couple $(sn_1, sn_2) \in Ch \times Ch$, $Dep(sn_1, sn_2) = \vee$. Moreover, $Deps$, where Dep is the dependency function and S is a subset of subnets, denotes the projection of Dep on the subset S . Finally, $Dep \setminus Deps$ denotes the dependency function obtained by eliminating $Deps$ from Dep .

The algorithm starts by waiting while the precondition of the current subnet is not satisfied (lines 1 – 3), i.e., the source places are not all marked. Then, line 4, the current subnet can be executed (leading to a sequence of firing sequence of transitions) before determining the set of subnets to be enabled in the next step *Init* (line 5). *Init* contains any subnet sn_i such that there is no other subnet s_j ($j \neq i$) such that $Dep(sn_j, sn_i) = \rightarrow$. If *Init* contains more than one element, it can be written as the following: $Init = ch_1 \cup \dots \cup ch_k \cup Par$ where ch_i , for $i = 1 \dots k$ are maximal sets containing exclusive subnets, *Par* contains concurrent (parallel) subnets (note that, in the first call, *Par* is necessarily empty), and $ch_i \cap ch_j = \emptyset$ and $ch_i \cap Par = \emptyset$, for any $i \neq j$. Lines 6 – 15 allows to keep in *Init* the subset *Par* and one representative of each subset ch_i and to update the set of subnets SN and the dependency function Dep . Once we have

Table 1. Dependency function of the decomposition of the shop process

	sn_1	sn_2	sn_3	sn_4	sn_5	sn_6	sn_7	sn_8	sn_9	sn_{10}	sn_{11}
sn_1	-	\rightarrow									
sn_2	-	-	\vee	\vee	\rightarrow	\vee	\vee	\vee	\vee	\vee	\rightarrow
sn_3	-	-	-	\rightarrow	\vee	\rightarrow	\rightarrow	\rightarrow	\rightarrow	\rightarrow	\rightarrow
sn_4	-	-	-	-	\vee	\rightarrow	\rightarrow	\rightarrow	\rightarrow	\rightarrow	\rightarrow
sn_5	-	-	-	-	-	\vee	\vee	\vee	\vee	\vee	\rightarrow
sn_6	-	-	-	-	-	-	\parallel	\rightarrow	\parallel	\rightarrow	\rightarrow
sn_7	-	-	-	-	-	-	-	\parallel	\rightarrow	\rightarrow	\rightarrow
sn_8	-	-	-	-	-	-	-	-	\parallel	\rightarrow	\rightarrow
sn_9	-	-	-	-	-	-	-	-	-	\rightarrow	\rightarrow
sn_{10}	-	-	-	-	-	-	-	-	-	-	\rightarrow

Algorithm 2 Execute-subnet

Require: Subnet $CurSN$, Set of subnets SN , Dependency function Dep

Ensure: Executing $CurSN$ and enabling next subnets

```

1: while Precondition of  $CurSN$  is not satisfied do
2:   wait;
3: end while
4:  $run(CurSN)$ 
5: Set of subnets  $Init = \{sn_i \mid \nexists sn_j : Dep(sn_j, sn_i) = \rightarrow\}$ 
6: if  $|Init| > 1$  then
7:   for all Choice  $ch \subseteq Init$  do
8:     Let  $sn_c$  be some chosen element of  $Ch$  (e.g., satisfying a given condition)
9:      $Init = (Init \setminus Ch) \cup \{sn_c\}$ 
10:     $Dep = Dep \setminus Dep_{Ch}$ 
11:     $SN = SN \setminus Ch$ 
12:   end for
13: end if
14:  $Dep = Dep \setminus Dep_{Init}$ 
15:  $SN = SN \setminus Init$ 
16: for all  $sn_i \in Init$  do
17:    $Execute\_subnet(sn_i, SN, Dep)$ 
18: end for

```

in $Init$ the set of the subnets to be concurrently enabled at the next step, each one will be enabled and receives the updated set of subnets and the dependency function (lines 16 – 18). Enabling a subnet is putting a token in the source place which coincides with the sink place of the current subnet. Thus, if the next subnet represents a synchronization between several subnets, it must wait until all these subnets finish their execution i.e., each source place is marked (the precondition becomes then satisfied).

To conclude, Algorithm 2 is associated to each subnet resulting from the decomposition of the whole WF-net and allows each subnet, executed separately, to autonomously launch the subnets to be executed in the next step. The first call is performed with an empty subnet unless the first enabled subnet is determined, a priori, by executing lines 5 – 15 in which case the first call can be performed by the subnet resulting from this execution.

Theorem 1. Let SN and Dep be respectively the set of subnets and the dependency function resulting from the application of Algorithm 1 on a WF-net N . Let σ be a sequence of transitions. σ is a firing sequence of N (i.e. $\sigma \in L(N)$) iff σ corresponds to an execution of Algorithm 2 with an empty subnet.

Proof

– \Rightarrow

Let σ be a run of SN and let us demonstrate, by induction on the length of σ ($|\sigma|$), that σ can be generated by Algorithm 2.

- $|\sigma| = 1$. Assume $\sigma = t$ and let i the source place of SN . Then, $t \in i^*$ and $m_0(i) > Pre(t, i)$ (where m_0 is the initial marking). Thus, the precondition of t is satisfied and the run of some of the subnets containing i (line 4) will allow to fire the transition t . Note that it is possible that i belongs to several subnets when, in SN , $|i^*| > 1$.
- Assume that any run of SN , of length $n \in \mathbb{N}$, can be generated by Algorithm 2. Let $\sigma = t_1 \dots t_{n+1}$ be a run of SN . Then, the sequence $t_1 \dots t_n$ can be generated by Algorithm 2. We distinguish the two following cases:
 1. t_n and t_{n+1} belong to the same subnet sn_i . In this case, the run of sn_i , by the instruction at line 4 allows to fire t_{n+1} directly after the firing of t_n .
 2. t_n and t_{n+1} belong to two different subnet sn_i and sn_j respectively. In this case, only one of the following conditions holds:
 - (a) $Dep(sn_i, sn_j) = \rightarrow$
 - (b) $Dep(sn_j, sn_i) = \rightarrow$
 - (c) $Dep(sn_i, sn_j) = ||$

Indeed, the value of $Dep(sn_i, sn_j)$ can not be \vee otherwise t_{n+1} would not be friable, within SN , after the firing of t_n .

- * The two first cases being symmetrical, assume that $Dep(sn_i, sn_j) = \rightarrow$. Then, there exists a set of subnets Par_i such that $\forall sn_k \in Par_i, \forall sn_l \in Par_i, Dep(sn_k, sn_l) = || \wedge Dep(sn_k, sn_j) = \rightarrow$. If such a set is empty let $Par_i = sn_i$. There exists $m < n$ s.t. for any subnet $sn_k \in Par_i$ there exists a run σ_k of sn_k s.t. $\sigma_{k, sn_k}^m = \sigma_k$ (where σ_m is the suffix of σ starting at the transition t_m and σ_{k, sn_k}^m denotes the projection of this run on the transitions of sn_k). Indeed if this does not hold, then t_{n+1} would not be enabled within SN . Thus, the precondition of sn_j becomes satisfied (line 1) as soon as t_n is fired and t_{n+1} is fired by the run of sn_j (line 4).
- * Assume now that $Dep(sn_i, sn_j) = ||$. Then, sn_i and sn_j can be launched in parallel by the loop instruction at line 16. One can then choose an interleaving allowing to fire t_{n+1} directly after t_n .

- \Leftarrow

Let $\sigma = t_1 \dots t_n$ be a sequence of transitions (execution) generated by Algorithm 2 and let us assume that σ is not a firing sequence of SN . Note first that t_1 is fireable at the initial marking of SN , otherwise the precondition (line 1) would not be satisfied and t_1 would not start any sequence generated by Algorithm 1. Thus, if σ is not a firing sequence of SN , then there exists $2 \leq i \leq n$ such that $t_1 \dots t_{i-1}$ is a run of SN and t_i is not fireable by the marking reached by this run.

We distinguish the two following cases:

1. t_{i-1} and t_i belong both to a subnet sn_i . In this case, t_i is not enabled by SN is not possible since instruction 4 launch the subnet sn_i which has the control on all its transitions i.e. the firing of t_i depends only on the firing of t_{i-1} .
2. t_{i-1} and t_i belong to sn_k and sn_l ($k = l$) respectively. t_i is not friable means that there exists a place $p_i \in \bullet t_i$ whose marking, after the firing of t_{i-1} , is strictly less than $Pre(t_i, p_i)$. This is not possible because the fact that t_i is generated by the algorithm (line 4) means that the precondition of the subnet sn_l is satisfied i.e., all the input places of t_i have been sufficiently marked by the execution of the loop at line 16.

3.4 Generating the Services' Code

In this step, we generate a Java code corresponding to each sliced service. The source code generation is based on BPEL to Java rules that we have defined (e.g. an *Invoke:one way* BPEL activity will be transformed to a Java asynchronous remote call, an *Invoke* activity to a Java synchronous remote call, an *Assign* activity to a Java assignment instruction and so on). For the *Invoke* activity, we add to services code a parameterized generic client stemming from a generic soap client code [27]. The parameters provided to each client are based on information on the *.bpel* descriptor and *.wsdl* descriptors corresponding to the process partner links. Furthermore, the structured BPEL activities (e.g. *IF*, *Else*, *While*, *Switch*, *Case*, etc.) still the same in Java.

However, before the BPEL to Java generation step, a correspondence task between Petri Net and BPEL must be done. This correspondence is based on the following fundamentals: (1) Each Petri Net transition is equivalent to its correspondent BPEL activity and (2) None of the Petri net places has equivalent in BPEL. Places make sense only in a Petri Net network.

In addition to that, it should be noted that some information are likely to be lost when we generate the Petri net from the *.bpel* descriptor. Indeed, the choice of execution branch is a non-deterministic choice in a Petri net. Thus, structured BPEL activities (e.g. *IF*, *While*, etc.) can not be represented using a Petri net. To find this kind of information, we annotate the Petri Net elements with these tests and/or conditions.

3.5 Packaging Services into Micro-containers

To perform this step, we reuse our previous work regarding elementary service deployment [19]. This work was based on the use of a specific prototype of service engines, called service Micro-Containers, that we have developed to address classical service containers scalability drawbacks in Cloud environments [11] [20].

Deploying services on existing PaaS involves four steps:

- Identifying the needed platform resources,
- Instantiating and initializing these resources,
- Uploading the service resources (e.g. source archives, libraries, deployment descriptor, etc.) on platform resources,
- Instantiating and initializing the service.

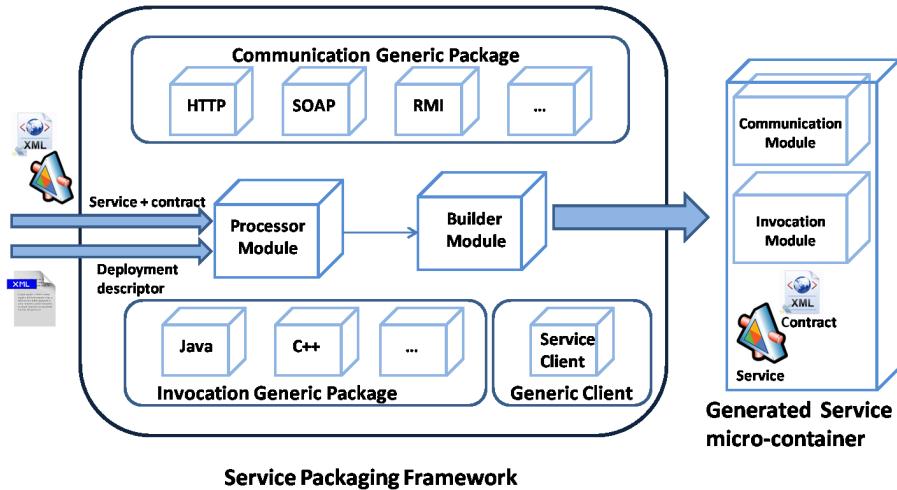


Fig. 4. Packaging framework architecture and building micro-container steps

When the service provider requires a specific hosting platform, we developed a standalone packaging and deployment framework to adopt these steps. Otherwise, deploying a service on our performed packaging framework consists of:

- Identifying the needed platform resources,
- Packaging these resources with the service to be deployed,
- Instantiating and initializing the platform and service resources.

This way of doing, consists of instantiate dedicating resources for each service to be deployed. This service and its platform resources are packaged in a quite service container that we called micro-container. Once the micro-container is packaged and built around the service, the developer can deploy it in a given target Cloud platform.

To build a service micro-container, one must mainly provide two elements (See Fig. 4): (1) The service with all its components (e.g. source archives, libraries, configuration files, etc.), and (2) A deployment descriptor that specifies the container options (e.g. main class, start command, etc.).

The processing module analyzes then the sources, detects the service binding types and associates to the service code a communication module implementing these bindings and a parameterized generic invocation module to run the service. The resulting code represents the generated micro-container code. micro-containers are specific types

of containers. They are only composed of the necessary modules for the deployed service, no more, no less. Generated micro-container hosts the service and implements its bindings regarding its supported communication protocols as long as they are included in the generic communication package in the packaging framework. They contains also a service invocation module supporting the service programming language as long as it is included in the generic invocation package in the packaging framework. It should be noted that before packaging time, the developer may also request to include some additional features to the micro-container if needed such as mobility [21] and/or monitoring [24].

Subjecting generated services from the shop process to the packaging framework allow us to package each one of them in a specific micro-container. The obtained set of the micro-containers are standalone applications (i.e. Java ARchive files) that can be run and interact to provide the initial business functionality of the BPEL process.

3.6 Deploying the Micro-Containers

There are two ways to deploy the obtained micro-containers. The first one consists of deploying them as standalone applications in a Cloud infrastructure (IaaS). To perform this, we upload and run the micro-containers in virtual machines instantiated from an infrastructure manager solution such as OpenNabula or OpenStack.

The second way to deploy the service-micro containers consists on the use of our already performed generic PaaS application provisioning and management API (called COAPS API) [12] [34] to deploy our service micro-containers in existing Cloud platforms (PaaS). COAPS ensures the real independence of our approach regarding the target PaaS. We have implemented COAPS to allow human and/or software agents to provision and manage PaaS applications. This API exposes a set of generic and RESTful HTTP operations for Cloud applications management and provisioning regarding the target PaaS. it provides an abstraction layer for existing PaaS allowing PaaS application provisioning in a unified manner.

To validate our achievement, we deployed our micro-containers generated from the shop process in Cloud Foundry PaaS using COAPS API. To perform this, we use the COAPS Web client which acts as an access point for the API implementations and allows a user to call the provisioning operations.

To execute the shop process, the developer have only to invoke the first generated service hosted in the first micro-container using a client that we have developed and is in charge of disseminating requests. The call sequence between the different deployed micro-containers ensures the semantic functionality of the initial process and returns the same execution result. Thus, we managed to deploy a BPEL in Cloud Foundry which is typically not able to host and provision appropriate resources for such applications.

4 Related Work

The deployment of service-based applications in the Cloud requires the provisioning of specific frameworks (e.g. Apache ODE in the case of business processes, etc.). However, existing PaaS providers provision only a static set of preinstalled resources and

therefore cannot satisfy complex requirements to provision appropriate frameworks for such applications. For this reason, some PaaS providers compensate the adhesion to specific hosting frameworks by continuous development of extensions requested by clients (for proprietary platforms) or proposed by users (for open-source platforms). But, this extension task is quite complex. It is a development task rather than an integration facility that consists of adding new components without hard coding.

A lot of works focused on deploying business processes and choreography-based process in the Cloud. In [14], the authors draw up an inventory of the different delivery models available to execute a BPEL process at IaaS, PaaS and SaaS layers. The conclusion made by the authors stipulates that there is still a lot of work to support BPEL provisioning especially in terms of communication processing between the different activities of the process. In addition to that, some works have proposed to transform and slice business processes to equivalent processes in order to be able to deploy them (or a part of them) in the Cloud [15] [16] [14].

In addition to that, several PaaS providers begin to integrate features to design, deploy and execute business processes. For example, Amazon Web Service proposes the Amazon Simple Workflow Service (Amazon SWF) which assists developers to coordinate the various processing steps in the process to deploy and to manage distributed execution state [13]. Force.com introduced a “Visual Process Manager” [13] and WSO2, which is based on Apache ODE process engine, provides a variant of its business process server as-a-Service [13]. However, all these features still proprietary and specific per provider. Indeed, each one of them imposes a specific deployment scenario and an interaction with the proprietary provider API. These features are extensions of the provider predefined list of containers rather than the platform support of business process execution and hosting.

In [18], the authors introduced a Self-Management Framework (SMF for short) to assist developers at deployment task. SMF handles interactions between the developer and the target PaaS API in order to install, configure and solve dependencies and conflicts issues related to the hosting application’s framework. Moreover, some PaaS providers, such as Cloud Foundry, propose a feature for deploying the required framework and/or specific platform resources (e.g. a proprietary service container instance) before deploying the application [17]. Nevertheless, these approaches delegates the framework installation and configuration to the developer which complicates significantly the deployment task, constitutes a step backward and is inconsistent with the PaaS business model as it has been defined.

5 Conclusion

In this paper, we present an approach to provision appropriate platform resources in order to provision BPEL processes in Cloud providers. Our approach consists of (1) Transforming the BPEL in an equivalent Petri Net, (2) Slice the Petri Net into a set of sub nets according to a slicing algorithm that we have defined. Each sub net represents an elementary service of the initial process. Then, (3) we generate the source code of each one of the sliced services following BPEL to Java transformation rules that we have defined as part of our work. Finally, (4) we package obtained services on our

already developed service micro-containers and (5) Deploy the micro-containers in a target Cloud provider. For the slicing, we defined algorithms to slice their correspondent Petri net into a set of dependent WF-nets and to determine the orchestration to follow for their execution. We also provided the proof of preservation of initial business process semantics when executing the WF-nets. To illustrate and show the feasibility of our proposition, we provide a realistic use case scenario, i.e. Shop process deployment in Cloud Foundry PaaS.

In the near future, we plan to conduct experimentations to evaluate performances when using our approach versus using classical business process engines such as Apache ODE. We also plan to consider BPEL processes elasticity support in the Cloud. The slicing algorithm that we have defined can be applied to a deployed BPEL process in order to refine the BPEL elasticity at activity level instead of the whole process.

Acknowledgment. The work presented in this paper was partially supported by the French Open-PaaS project.

References

1. Hoenisch, P., Schulte, S., Dustdar, S., Venugopal, S.: Self-Adaptive Resource Allocation for Elastic Process. In: Proceedings of IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA (2013)
2. Foster, I., Zhao, Y., Raicu, I., Shiyoung, L.: Cloud Computing and Grid Computing 360-Degree Compared. In: Proceedings of Grid Computing Environments Workshop, Austin, USA, pp. 1–10 (2008)
3. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology (NIST). Special Publication 800-145 (Draft). U.S Department of Commerce, USA (2013)
4. Service Component Architecture. Advancing open standards for the information society (OASIS) Open CSA (2014), <http://oasis-opencsa.org/sca>
5. OASIS Web Services Business Process Execution Language (BPEL). Advancing open standards for the information society, OASIS (2014), <https://www.oasis-open.org/committees/wsbpel/>
6. Business Process Model and Notation V2.0, Object Management Group (OMG), Technical Report (2011), <http://taval.de/publications/BPMN20>
7. Rimal, B.P., Eunmi, C., Lumb, I.: A taxonomy and survey of cloud computing systems. In: Proceedings of the International Joint Conference on INC, IMS and IDC, Seoul, Korea, pp. 44–51 (2009)
8. Fu, X., Bultan, T., Su, J.: Analysis of interacting BPEL web services. In: Proceedings of the International Conference on World Wide Web, NY, USA, pp. 621–630 (2004)
9. Yangui, S., Ben Nasrallah, M., Tata, S.: PaaS-independent approach to provision appropriate Cloud resources for SCA-based applications deployment. In: Proceedings of the International Conference on Semantics, Knowledge & Grids, Beijing, China (2013)
10. Yangui, S., Tata, S.: Paas Elements for Hosting Service-based Applications. In: Proceedings of the International Conference on Cloud Computing and Services Science, Porto, Portugal, pp. 476–479 (2012)
11. Yangui, S., Mohamed, M., Tata, S., Moalla, S.: Scalable Service Containers. In: Proceedings of the IEEE International Conference on Cloud Computing Technology and Science, Athenes, Greece, pp. 348–356 (2011)

12. Sellami, M., Yangui, S., Mohamed, M., Tata, S.: PaaS-independent Provisioning and Management of Applications in the Cloud. In: Proceedings of IEEE International Conference on Cloud Computing, Santa Clara Marriott, USA, pp. 693–700 (2013)
13. The Amazon Simple Workflow Service developer guide (2014), <http://docs.aws.amazon.com/amazonswf/latest/developerguide/swf-welcome.html>
14. Anstett, T., Leymann, F., Mietzner, R., Strauch, S.: Towards BPEL in the Cloud: Exploiting Different Delivery Models for the Execution of Business Processes. In: Proceedings of World Conference on Services-I, Los Angeles, USA, pp. 670–677 (2009)
15. Dornemann, T., Juhnke, E., Freisleben, B.: On-Demand Resource Provisioning for BPEL Workflows Using Amazon’s Elastic Compute Cloud. In: Proceedings of International Symposium on Cluster Computing and the Grid, Shanghai, China, pp. 140–147 (2009)
16. Wagner, S., Kopp, O., Leymann, F.: Towards choreography-based process distribution in the cloud. In: Proceedings of Cloud Computing and Intelligence Systems, Beijing, China, pp. 490–494 (2011)
17. Cloud Foundry official blog. Deploying a service container on CF using the standalone framework (2014), <http://blog.cloudfoundry.com/2012/06/18/deploying-tomcat-7-using-the-standalone-framework/>
18. Wei, H., Shao, J., Liu, B., Liu, H., Wang, Q., Mei, H.: A Self-management Approach for Service Developers of PaaS. In: Proceedings of the IEEE International Symposium on Service Oriented System Engineering, Irvine, Canada, pp. 85–92 (2011)
19. Yangui, S., Tata, S.: CloudServ: PaaS Resources Provisioning for Service-Based Applications. In: Proceedings of IEEE International Conference on Advanced Information Networking and Applications, Barcelona, Spain, pp. 522–529 (2013)
20. Mohamed, M., Yangui, S., Moalla, S., Tata, S.: Service micro-container for service-based applications in Cloud environments. In: Proceedings of the IEEE International Conference on Collaboration Technologies and Infrastructures, Paris, France, pp. 61–66 (2011)
21. Omezine, A., Yangui, S., Bellamine, N., Tata, S.: Mobile Service micro-container for Cloud environments. In: Proceedings of the IEEE International Conference on Collaboration Technologies and Infrastructures, Toulouse, France, pp. 154–160 (2012)
22. Mohamed, M., Belaid, D., Tata, S.: Monitoring and Reconfiguration for OCCI Resources. In: IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2013 to be held in Bristol, UK (2013)
23. Amziani, M., Klai, K., Melliti, T., Tata, S.: Time-based Evaluation of Service-based Business Process Elasticity in the Cloud. In: IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2013 to be held in Bristol, UK (2013)
24. Mohamed, M., Belaid, D., Tata, S.: How to Provide Monitoring Facilities to Services When They Are Deployed in the Cloud? In: Proceedings of the International Conference on Cloud Computing and Services Science, Porto, Portugal, 258–263 (2012)
25. Hinz, S., Schmidt, K., Stahl, C.: Transforming BPEL to Petri Nets. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 220–235. Springer, Heidelberg (2005)
26. Stahl, C.: A Petri Net Semantics for BPEL. Informatik-Berichte 188, Humboldt-Universität zu Berlin (2005)
27. A WSDL Generic Soap Client (2014), <https://github.com/impactcentre/interoperability-framework/tree/master/interfaces/web/generic-soap-client>
28. Lohmann, N., Verbeek, E., Ouyang, C., Stahl, C.: Comparing and evaluating Petri net semantics for BPEL. International Journal of Business Process Integration and Management 4(1), 60–73 (2009)
29. Schmidt, K.: LoLA A Low Level Analyser. In: Nielsen, M., Simpson, D. (eds.) ICATPN 2000. LNCS, vol. 1825, pp. 465–474. Springer, Heidelberg (2000)

30. LoLA tool Website (2014), <http://www2.informatik.hu-berlin.de/top/lola/lola.html>
31. Billington, J., et al.: The Petri Net Markup Language: Concepts, Technology, and Tools. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 483–505. Springer, Heidelberg (2003)
32. Dijkman, R., Dumas, M., Ouyang, C.: Semantics and Analysis of Business Process Models in BPMN. *Information and Software Technology* 50, 1281–1294 (2008)
33. Wil, M., van der Aalst, P.: The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers* 8, 21–66 (1998)
34. Yangui, S., Marshall, I.J., Laisne, J.P., Tata, S.: CompatibleOne: The Open Source Cloud Broker. *Journal of Grid Computing* 12(1), 93–109 (2014)

Log-Based Understanding of Business Processes through Temporal Logic Query Checking

Margus Räim¹, Claudio Di Ciccio², Fabrizio Maria Maggi¹,
Massimo Mecella³, and Jan Mendling^{2,*}

¹ University of Tartu, Estonia
f.m.maggi@ut.ee

² Vienna University of Economics and Business, Austria
{claudio.di.ciccio,jan.mendling}@wu.ac.at

³ Sapienza Università di Roma, Italy
massimo.mecella@uniroma1.it

Abstract. Process mining is a discipline that aims at discovering, monitoring and improving real-life processes by extracting knowledge from event logs. Process discovery and conformance checking are the two main process mining tasks. Process discovery techniques can be used to learn a process model from example traces in an event log, whereas the goal of conformance checking is to compare the observed behavior in the event log with the modeled behavior. In this paper, we propose an approach based on *temporal logic query checking*, which is in the middle between process discovery and conformance checking. It can be used to discover those LTL-based business rules that are valid in the log, by checking against the log a (user-defined) class of rules. The proposed approach is not limited to provide a boolean answer about the validity of a business rule in the log, but it rather provides valuable diagnostics in terms of traces in which the rule is satisfied (witnesses) and traces in which the rule is violated (counterexamples). We have implemented our approach as a proof of concept and conducted a wide experimentation using both synthetic and real-life logs.

Keywords: Process Discovery, Business Rules, Linear Temporal Logic, Temporal Logic Query Checking.

1 Introduction

The increasing availability of event data has stimulated the uptake of automatic process discovery in research and practice. Discovery techniques typically take event logs as an input to generate a process model as an output, for instance, in the form of a Petri net. This works particularly well for structured processes.

However, event logs might not always stem from structured processes, but from RFID readers, from GPS data of trucks, or from unstructured knowledge-intensive processes. Events from these types of processes have often in common that they yield

* The work of Massimo Mecella has been partly supported by the Italian projects SUPER, RoMA and the cluster Social Museum and Smart Tourism, and by the EU projects SmartVortex and VOICE. The work of Claudio Di Ciccio and Jan Mendling has received funding from the EU Seventh Framework Programme under grant agreement 318275 (GET Service).

spaghetti-like Petri net models. In order to make discovery work for those event logs, declarative representations have been defined that capture behavioral constraints.

So far, several specifications of declarative behavior have been defined, e.g., Declare [23], behavioral profiles [26], or Dynamic Condition Response Graphs [16]. A standard formalism underneath these notations is Linear Temporal Logic (LTL). However, existing LTL-based discovery techniques like [12,14,20,22,19] hardly make use of the flexibility that LTL provides, but rather work with a limited set of predefined templates. On the other hand, works like [25] for conformance checking with respect to LTL-based business rules, do not support domain experts in finding classes of LTL rules that are interesting to check. In this paper, we address this research gap. Our contribution is an approach for discovering LTL rules from event logs based on temporal logic query checking using placeholders. The technique produces as outcome activities that can replace the placeholders in the query to produce rules satisfied in the log. If the activities considered for these placeholders cover the full process alphabet, the approach complements existing process discovery techniques. If a placeholder is replaced by a single activity, our approach supports conformance checking. In the middle there is the possibility of choosing the replacement for a placeholder within a (user-defined) set of activities. The proposed approach is not limited to provide a boolean answer about the validity of a business rule in the log, but it rather provides diagnostics in terms of traces in which the solution is satisfied (*witnesses*) and traces in which the solution is violated (*counterexamples*). Our concepts have been implemented and evaluated using synthetic and real-life logs. In this way, we are able to find behavioral rules that might not have been visible, e.g., to predefined Declare templates.

Against this background, the paper is structured as follows. Section 2 defines the preliminaries and the research problem. Section 3 defines our approach based on the log temporal structure and a corresponding query checking algorithm. Section 4 describes experimental results of using our implementation on synthetic and real-life logs. Section 5 reflects our contribution in the light of related work. Finally, Section 6 concludes the paper.

2 Preliminaries

The research objective of this work is to support the domain expert with a technique to discover temporal business rules and to define rule templates with placeholders. This is, on the one hand, a verification problem [3], and, on the other hand, a temporal query checking problem. We approach this from the perspective of temporal logical query checking; therefore we first introduce the notions of event logs, linear temporal logic over finite traces and respective queries, and then we discuss the research problem.

2.1 Event Logs

An event log captures the manifestation of events pertaining to the instances of a single process. A *process instance* is also referred to as a *case*. Each event in the log corresponds to a single case and can be related to an activity or a task. Events within a case need to be *ordered*. An event may also carry optional additional information like

timestamp, transaction type, resource, costs, etc. For analysis, we need a function that maps any event e onto its *class* \bar{e} . In this paper, we assume that each event is classified based on its activity. \mathcal{A} denotes the set of *activities* (*log alphabet*). \mathcal{A}^+ is the set of all non-empty finite sequences. A case is described as a *trace* over \mathcal{A} , i.e., a finite sequence of activities.

Definition 1 (Trace). Given a finite log alphabet \mathcal{A} of events, the trace $t = \langle a_1, \dots, a_n \rangle$ is a finite sequence of events $a_i \in \mathcal{A}$. We denote by $t(i)$ the i -th event in the trace, with $i \in [1, n]$ where n is the length of t , i.e., $n = |t|$. Thus, $t \in \mathcal{A}^+$, where \mathcal{A}^+ denotes the set of non-empty sequences of elements of \mathcal{A} . Two traces t and t' are defined equivalent when $|t| = |t'| = n$ and, for all $i \in [1, n]$, $t(i) = t'(i)$.

Examples of traces are $t_1 = \langle a, b, c, d \rangle$ and $t_2 = \langle a, b, b, b, a, d \rangle$.

Definition 2 (Event Log). An event log $L \in \mathcal{P}(\mathcal{A}^+)$ is a finite collection of traces.

For example, $L = [\langle a, b, c, d \rangle, \langle a, b, c, d \rangle, \langle a, b, b, b, a, d \rangle]$ is a log consisting of three cases. Two cases follow the trace $\langle a, b, c, d \rangle$ and one case follows the trace $\langle a, b, b, b, a, d \rangle$.

2.2 Linear Temporal Logic over Finite Traces

Linear Temporal Logic (LTL) [24] is a language meant to express properties that hold true in systems that change their state over time. The state of the system is expressed in terms of propositional formulas. The evolution is defined by transitions between states. A typical LTL query expressing *fairness* conditions is $\square\Diamond\Phi$, where Φ is a propositional formula indicating the condition to always (\square) eventually (\Diamond) hold true. LTL_f [10,9] is the variant of LTL interpreted over finite system executions. It adopts the syntax of LTL. Formulas of LTL (and LTL_f) are built from a set \mathcal{A} of propositional symbols (*atomic propositions*, altogether constituting the *alphabet*) and are closed under the boolean connectives (\neg , unary, and \vee , \wedge , \rightarrow , \leftrightarrow , binary) and the temporal operators \circ (*next*), \Diamond (*eventually*), \square (*always*), unary, and U (*until*), binary. The syntax is defined as follows.

$$\begin{aligned} \varphi, \psi &= a &|& \rho &|& \tau && (\text{with } a \in \mathcal{A}) \\ \rho &= \neg\varphi &|& \varphi \vee \psi &|& \varphi \wedge \psi &|& \varphi \rightarrow \psi &|& \varphi \leftrightarrow \psi \\ \tau &= \circ\varphi &|& \Diamond\varphi &|& \Box\varphi &|& \varphi U \psi \end{aligned}$$

Intuitively, $\circ\varphi$ means that φ holds true in the next instant in time, $\Diamond\varphi$ signifies that φ holds eventually before the last instant in time (included), $\Box\varphi$ expresses the fact that from the current state until the last instant in time φ holds, $\varphi U \psi$ says that ψ holds eventually in the future and φ holds until that point. The semantics of LTL_f is provided in terms of finite runs,¹ i.e., finite sequences of consecutive instants in time, represented by finite words π over the alphabet $2^{\mathcal{A}}$. The instant i in run π is denoted as $\pi(i)$, with $i \in [1, |\pi|]$, $|\pi|$ being the length of the run. We remark here that the alphabet

¹ Called “traces” in the literature [10], here renamed in order to avoid confusions with the sequences of events in logs.

of finite words for LTL_f runs do not correspond to \mathcal{A} , but to any possible propositional interpretation of the propositional symbols in \mathcal{A} ($2^{\mathcal{A}}$). In the following, we indicate that, e.g., a is interpreted as true (\top) at instant i in π by $a \in \pi(i)$. Conversely, if $a \notin \pi(i)$, a is interpreted as false (\perp). Given a finite run π , we inductively define when an LTL_f formula φ (resp. ψ) is true at an instant i , denoted as $\pi, i \models \varphi$ (resp., $\pi, i \models \psi$), as:

$$\pi, i \models a \text{ for } a \in \mathcal{A}, \text{ iff } a \in \pi(i) \text{ (} a \text{ is interpreted as true in } \pi(i)\text{);}$$

$$\pi, i \models \neg\varphi \text{ iff } \varphi, i \not\models \pi(i);$$

$$\pi, i \models \varphi \wedge \psi \text{ iff } \pi, i \models \varphi \text{ and } \pi, i \models \psi;$$

$$\pi, i \models \varphi \vee \psi \text{ iff } \pi, i \models \varphi \text{ or } \pi, i \models \psi;$$

$$\pi, i \models \circ\varphi \text{ iff } \pi, i+1 \models \varphi, \text{ having } i < |\pi|;$$

$$\pi, i \models \varphi U \psi \text{ iff for some } j \in [i, |\pi|], \text{ we have that } \pi, j \models \psi, \text{ and for all } k \in [i, j-1], \\ \text{we have that } \pi, k \models \varphi.$$

The semantics of remaining operators can be derived by recalling that:

$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi);$$

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi);$$

$$\Diamond\varphi \equiv \top U \varphi;$$

$$\Box\varphi \equiv \neg\Diamond\neg\varphi.$$

2.3 Temporal Logic Query Checking

Model checking [8] was originally proposed as a verification technique: given a model of a software system and a specification, formal methods are adopted to prove the compliance of the model to the specification. Temporal model checking focuses on the automated verification of temporal logic formulas over temporal structures, modeled as Kripke structures. Kripke structures are finite state automata having sets of states, a labeling function mapping states to atomic propositions interpreted as true, and a left-total transition relation among states, i.e., every state leads to a transition. Initial states are those states from which runs of the model enactment can start.

Temporal logic query checking [3] aims at discovering properties of the model which are not known *a priori*. Therefore, the extra input element as compared to model checking is a *query*, i.e., a temporal logic formula containing so-called *placeholders*, typically denoted as $?_x$. The output of the query checking technique is a set of temporal logic formulas, which derive from the input query. Every output formula stems from the replacement of placeholders by propositional formulas, which make the overall temporal logic formula satisfied in the given Kripke structure.

The seminal work of Chan considered Computation Tree Logic (CTL) [7] as the language for expressing queries. Unlike LTL, which adopts a linear time assumption, CTL is a branching-time logic: the evaluation of the formula is based on a tree-like structure, where different evolutions of the system over time are simultaneously considered in different branches of the computation model. Consider as an example the CTL query $AG(?_x \vee p)$. This query states that for every evolution of the system (A), it is globally true (G) that p or $?_x$ hold. In order to drive the verification by limiting the search space of possible satisfying replacements for placeholders, Chan introduced the opportunity to limit the propositional symbols involved. For instance, a CTL query like $AG(?_x)\{a, b\}$ would restrict the possible solutions returned to the ones including only a and b as proposition letters.

3 Proposed Approach

In this section, we describe the proposed approach in detail. The starting point is an event log and an LTL_f query. The outcome is a set of formulas derived from the query by placeholder replacement, along with diagnostics about the validity of each formula in the input log.

3.1 From the Event Log to a Log Temporal Structure

Here, we illustrate the steps to obtain from the log a behaviorally equivalent temporal structure, for evaluating LTL_f formulas over it. The first step consists in creating a trace temporal structure for each trace in the log. The only allowed run for such a temporal structure is the trace itself. Then, we combine the collection of trace temporal structures into a single entity (community). Thereupon, we derive a log temporal structure, bisimilar to such a community, by joining the common suffixes of traces. This translates to joining the states of trace temporal structures for which all next successor states are pairwise interpreted as the same atomic propositions. The resulting polytree is the input for the query checking algorithm described later in Section 3.2.

Definition 3 (Trace temporal structure). A trace temporal structure is a tuple $\mathcal{T} = \langle S, \triangleright, s_1, s_f, L \rangle$ over alphabet \mathcal{A} where:

S is a finite non-empty set of states.

$s_1, s_f \in S$ are the initial and final state of \mathcal{T} .

$L : S \rightarrow (\mathcal{A} \rightarrow \{\top, \perp\})$ is the labeling function, associating each state $s \in S$ to an interpretation of each propositional literal $a \in \mathcal{A}$, either assigned as true, \top , or false, \perp . If, e.g., a is interpreted as true in state s , we indicate it as follows: $L(s, a) \mapsto \top$.

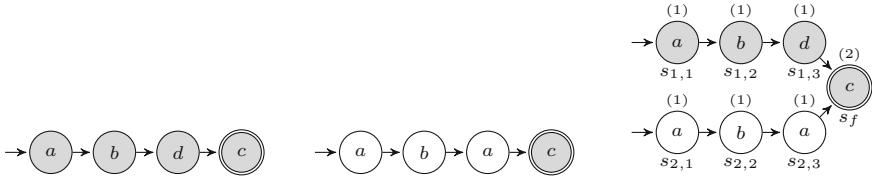
$\triangleright : S \setminus \{s_f\} \rightarrow S$ is the bijective successor state function, associating each state $s \in S \setminus \{s_f\}$ to one following state $s' \in S$. For the sake of conciseness, when $(s, s') \in \triangleright$, we also refer to s' as $s\triangleright$.

The successor state function \triangleright constitutes the transition relation for the temporal structure. It is bijective over the domain of $S \setminus \{s_f\}$ and the codomain of S . As such, an inverse function exists, which we refer to as \triangleleft (predecessor state function). When $(s, s') \in \triangleright$, we also refer to s as $\triangleleft s'$. With a slight abuse of notation, we define predecessors and successors for sets of states too: given $\hat{S} \subseteq S$, we have $\triangleleft \hat{S} \doteq \{s \in S | s = \triangleleft \hat{s}, \hat{s} \in \hat{S}\}$ and $\hat{S}\triangleright \doteq \{s \in S | s = \hat{s}\triangleright, \hat{s} \in \hat{S}\}$. We denote by $\mathcal{A}^{\mathcal{T}}$ the universe of possible trace temporal structures over alphabet \mathcal{A} .

Figs. 1a and 1b show two trace temporal structures. They stem from two traces, i.e., $\langle a, b, d, c \rangle$ and $\langle a, b, a, c \rangle$. Indeed, in order to check LTL_f formulas on traces, we apply a mapping function \mathcal{M} , defined as follows. The \mathcal{M} -mapping of an element of \mathbf{t} to an element of \mathcal{T} is denoted by the infix notation $\xrightarrow{\mathcal{M}}$.

Definition 4 (\mathcal{M} -mapping from a trace to a trace temporal structure). Let $\mathbf{t} = \langle a_1, \dots, a_n \rangle \in \mathcal{A}^+$ be a trace, and $\mathcal{T} = \langle S, \triangleright, s_1, s_f, L \rangle \in \mathcal{A}^{\mathcal{T}}$ a trace temporal structure, both defined over the common alphabet \mathcal{A} .

A mapping $\mathcal{M} : \mathcal{A}^+ \rightarrow \mathcal{A}^{\mathcal{T}}$ is such that:



- (a) States satisfying $\square(a \rightarrow \Diamond b)$ in trace temporal structure \mathcal{T}_1 , derived from $\mathbf{t}_1 = \langle a, b, d, c \rangle$.
(b) States satisfying $\square(a \rightarrow \Diamond b)$ in trace temporal structure \mathcal{T}_2 , derived from $\mathbf{t}_2 = \langle a, b, a, c \rangle$.
(c) States satisfying $\square(a \rightarrow \Diamond b)$ in log temporal structure \mathcal{L} for log L , consisting of \mathbf{t}_1 and \mathbf{t}_2 .

Fig. 1. Trace temporal structures and log temporal structure for $\log L = [\langle a, b, d, c \rangle, \langle a, b, a, c \rangle]$ and evaluation of formula $\square(a \rightarrow \Diamond b)$ over them. In Fig. 1c, weights are specified among parentheses above states. State identifiers are below states.

- for all $i \in [1, n]$, $\mathbf{t}(i) \xrightarrow{\mathcal{M}} s_i$, with $s_i \in S$;
- for all $s_i \in S$, there exists $i \in [1, n]$ such that $\mathbf{t}(i) \xrightarrow{\mathcal{M}} s_i$;
- given $i, j \in [1, n]$, if $i \neq j$ then $s_i \neq s_j$ (\mathcal{M} preserves bijection over events and states);
- given $a \in \mathcal{A}$ s.t. $a = \mathbf{t}(i)$, then $L(s, a) \mapsto \top$, and for all $a' \in \mathcal{A}$ s.t. $a' \neq a$, $L(s, a') \mapsto \perp$;
- $\mathbf{t}(1) \xrightarrow{\mathcal{M}} s_1$; $\mathbf{t}(n) \xrightarrow{\mathcal{M}} s_f$.

Fig. 1a and Fig. 1b show the trace temporal structures \mathcal{T}_1 and \mathcal{T}_2 to which $\mathbf{t}_1 = \langle a, b, d, c \rangle$ and $\mathbf{t}_2 = \langle a, b, a, c \rangle$ \mathcal{M} -map respectively. The reader can notice that each trace corresponds to the only allowed *run* of the \mathcal{M} -mapped temporal structure (see Section 2.2). As a log L is a collection of traces (see Section 2.1), we denote a collection of mapped temporal structures by $\mathcal{M}(L)$, with a slight abuse of notation. By definition, $\mathcal{M}(L) \subseteq \mathcal{A}^{\mathcal{T}}$. In order to evaluate the LTL_f formula over a representation of the entire log, seen as a whole, we introduce the following temporal structure.

Definition 5 (Log weighted temporal structure). A log weighted temporal structure (or log temporal structure, for short) is a tuple $\mathcal{L} = \langle S, \triangleright, S_1, S_f, L, W \rangle$ over alphabet \mathcal{A} where:

S is a finite non-empty set of states.

$S_1, S_f \in S$ are the initial and final sets of states in \mathcal{L} .

$L : S \rightarrow (\mathcal{A} \rightarrow \{\top, \perp\})$ is the labeling function, associating each state $s \in S$ to an interpretation of each propositional literal $a \in \mathcal{A}$.

$\triangleright : S \setminus S_f \rightarrow S$ is the surjective successor state function, which associates each state $s \in S \setminus S_f$ to one following state $s' \in S$.

$W : S \rightarrow \mathbb{N}^+$ is the weighing function, associating each state $s \in S$ to a positive integer $W(s) > 0$ (weight).

The successor state function \triangleright constitutes the transition relation for the temporal structure. Unlike the successor state function of trace temporal structures, the successor state function of log temporal structures is only surjective and not injective: in other

words, it can happen that $s \neq s'$ but $s\triangleright = s'\triangleright$, i.e., different states share the same successor. Therefore, no inverse function can be derived. Nonetheless, with a slight abuse of notation, we denote as $\triangleleft s'$ the set of states whose successor is $s' \in S$: $\triangleleft s' \doteq \{s \in S | s' = s\triangleright\}$. Fig. 1c shows the log temporal structure derived from the collections of temporal structures depicted in Figs. 1a and 1b. We denote the universe of possible log temporal structures defined over \mathcal{A} as $\mathcal{A}^{\mathcal{L}}$.

We introduce now the *community of trace temporal structures* \mathcal{C} , which is needed to pass from a collection of tuples to a single-tuple equivalent representation. It consists of the union of trace temporal structures. For the sake of clarity, we adopt the “.” symbol in the remainder to specify to which temporal structure a given element belongs to. For instance, the set of states of community \mathcal{C} is denoted as $\mathcal{C}.S$, whereas the set of states of trace temporal structure \mathcal{T}_i is denoted as $\mathcal{T}_i.S$. We define the community of trace temporal structures $\mathcal{C} = \langle S, \triangleright, S_1, S_f, L \rangle$, derived from a collection of trace temporal structures $\{\mathcal{T}_1, \dots, \mathcal{T}_m\}$, as follows:

$$\begin{aligned}\mathcal{C}.S &= \bigcup_{i=1}^m \mathcal{T}_i.S & \mathcal{C}.S_f &= \bigcup_{i=1}^m \{\mathcal{T}_i.s_f\} & \mathcal{C}.\triangleright &= \bigcup_{i=1}^m \mathcal{T}_i.\triangleright \\ \mathcal{C}.S_1 &= \bigcup_{i=1}^m \{\mathcal{T}_i.s_1\} & \mathcal{C}.L &= \bigcup_{i=1}^m \mathcal{T}_i.L\end{aligned}$$

We denote the universe of possible communities of trace temporal structures over \mathcal{A} as $\mathcal{A}^{\mathcal{C}}$. Based on the concept of community \mathcal{C} that jointly represents single traces $\mathcal{T}_i, \dots, \mathcal{T}_m \in \mathcal{M}(L)$, we can define the \mathcal{R} -mapping. The \mathcal{R} -mapping of an element of \mathcal{C} to an element of \mathcal{L} is denoted by the infix notation $\xrightarrow{\mathcal{R}}$.

Definition 6 (\mathcal{R} -mapping of a community of trace temporal structures to a log temporal structure). Let $\mathcal{C} = \langle S, \triangleright, S_1, S_f, L \rangle \in \mathcal{A}^{\mathcal{C}}$ be a community of trace temporal structures and $\mathcal{L} = \langle S, \triangleright, S_1, S_f, L, W \rangle \in \mathcal{A}^{\mathcal{L}}$ be a log temporal structure, defined over the common alphabet \mathcal{A} .

$\mathcal{R} : \mathcal{A}^{\mathcal{C}} \rightarrow \mathcal{A}^{\mathcal{L}}$ is such that:

- given a state $s_C \in \mathcal{C}.S$, and a state $s_L \in \mathcal{L}.S$,
 - $s_C \xrightarrow{\mathcal{R}} s_L$ iff
 - * $\mathcal{C}.L(s_C) = \mathcal{L}.L(s_L)$, i.e., the two states are interpreted in the same way over alphabet \mathcal{A} , and
 - * either $s_C \in \mathcal{C}.S_f$ (s_C is a final state of \mathcal{C}) or $(s_C\triangleright) \xrightarrow{\mathcal{R}} (s_L\triangleright)$, i.e., the successor state of s_C maps to the successor state of s_L ;
 - $W(s_L) = \left| \left\{ s_C \in \mathcal{C}.S | s_C \xrightarrow{\mathcal{R}} s_L \right\} \right|$, i.e., the weight of a state s_L in the log temporal structure \mathcal{L} is equal to the number of states in \mathcal{C} that map to s_L :
- given states $s_L, s'_L \in \mathcal{L}.S$ and $s_C, s'_C \in \mathcal{C}.S$, $(s_L, s'_L) \in \mathcal{L}.\triangleright$ iff $(s_C, s'_C) \in \mathcal{C}.\triangleright$;
- $s_{\mathcal{L}_f} \in \mathcal{L}.S_f$ iff $s_{\mathcal{C}_f} \in \mathcal{C}.S_f$ and $s_{\mathcal{C}_f} \xrightarrow{\mathcal{R}} s_{\mathcal{L}_f}$, i.e., final states in \mathcal{C} correspond to final states in \mathcal{L} ;
- $s_{\mathcal{L}_1} \in \mathcal{L}.S_1$ iff $s_{\mathcal{C}_1} \in \mathcal{C}.S_1$ and $s_{\mathcal{C}_1} \xrightarrow{\mathcal{R}} s_{\mathcal{L}_1}$, i.e., initial states in \mathcal{C} correspond to initial states in \mathcal{L} .

\mathcal{R} establishes a biunivocal correspondence for every element of \mathcal{C} and \mathcal{L} , except states. The technique that we adopt to obtain the \mathcal{R} -mapping log temporal structure \mathcal{L} from a

collection of trace temporal structures $\mathcal{M}(L)$, represented as a community \mathcal{C} , is depicted in Fig. 3. All final states that are associated by the labeling function to the same literal in \mathcal{A} are joint in one single final state. It is associated with the same shared literal and has a weight amounting to the sum of joined states. The state successor function is modified by linking the predecessors to the new joint state. Those operations are repeated step by step along the predecessors of the final states in the trace temporal structures, up to the initial ones.

The topology of the resulting log temporal structure is always an *inverted polytree*. We name it inverted to highlight that every subtree of the polytree has one root and multiple leaves only if we consider the *inverse* transition relation \triangleleft for the direction of arcs (see Fig. 3e).

We conclude this section with a theorem, allowing us to solve the problem of LTL_f query checking over event logs by analyzing the corresponding log temporal structure. We omit its proof due to space reasons.

Theorem 1. *If a community of trace temporal structures \mathcal{C} \mathcal{R} -maps to a log temporal structure \mathcal{T} , \mathcal{C} and \mathcal{T} are bisimilar.*

Temporal logics are bisimulation-invariant [11,8]. As a consequence, we can evaluate LTL_f formulas on the log temporal structure in order to show whether the corresponding business rules were satisfied or not in the log.

Corollary 1. *Given a log L , a community of trace temporal structures \mathcal{C} derived from $\mathcal{M}(L)$, and a log temporal structure \mathcal{L} s.t. \mathcal{C} \mathcal{R} -maps to it, the weight of an initial state s_1 of \mathcal{L} is equal to the number of equivalent traces $t, t', \dots, t^m \in L$ s.t. $t(1) = t'(1) = \dots = t^m(1) = L(s_1)$.*

For space reasons, we omit the proof here. However, it can be verified by considering the construction of the log temporal structure. An example is given in Fig. 3, where Fig. 3a and Fig. 3f highlight the states satisfying the same formula.

3.2 Query Checking Algorithm

Algorithm 1 describes the main procedure of our approach. The *check* procedure takes as input the log temporal structure \mathcal{L} (obtained from a log through the steps

Algorithm 1. *check($\mathcal{L}, \Phi(?_{x_1}\{\mathcal{A}_{x_1}\}, \dots, ?_{x_n}\{\mathcal{A}_{x_n}\}), \mathcal{A}$), checking LTL_f query $\Phi(?_{x_1}\{\mathcal{A}_{x_1}\}, \dots, ?_{x_n}\{\mathcal{A}_{x_n}\})$ on temporal structure \mathcal{L} , defined over alphabet \mathcal{A}*

Input: $\mathcal{L} = \langle S, \triangleright, S_1, S_f, L, W \rangle$, LTL_f query $\Phi(?_{x_1}\{\mathcal{A}_{x_1}\}, \dots, ?_{x_n}\{\mathcal{A}_{x_n}\})$ with placeholders $?_{x_1} \dots ?_{x_n}$ resp. bounded to literals $\mathcal{A}_{x_1} \dots, \mathcal{A}_{x_n}$, subsets of alphabet \mathcal{A}
Output: $\mathcal{B}_\Phi^{\hat{S}} = \left\{ \langle \hat{S}_\Phi, \Phi \rangle \right\}^*$, set of initial states of witnesses (\hat{S}_Φ) for each LTL_f formula Φ , obtained through the replacement of placeholders with the bounded literals.

```

1  $\mathcal{B}_\Phi^{\hat{S}} : \emptyset$ 
2 foreach  $\{a_1, \dots, a_n\} \in (\mathcal{A}_{x_1} \times \dots \times \mathcal{A}_{x_n})$  do
3    $\left| \begin{array}{l} \Phi : \Phi(a_1, \dots, a_n) \\ \mathcal{B}_\Phi^{\hat{S}} : \mathcal{B}_\Phi^{\hat{S}} \cup \langle evalForm(\mathcal{L}, \Phi, S, \mathcal{A}), \Phi \rangle \end{array} \right.$ 
4
5 return  $\mathcal{B}_\Phi^{\hat{S}}$ 

```

**Fig. 2.** Evaluation trees

described in Section 3.1) and the query expressing the business rule template, denoted as $\Phi(?_{x_1}\{\mathcal{A}_{x_1}\}, \dots, ?_{x_n}\{\mathcal{A}_{x_n}\})$. To each placeholder $?_{x_i}$ a set of symbols \mathcal{A}_{x_i} is associated, restricting the possible assignments of placeholders. The sets of symbols are contained in the log alphabet \mathcal{A} . The output is a set of tuples $\langle \hat{S}_\Phi, \Phi \rangle$, where \hat{S}_Φ is the set of states in \mathcal{L} satisfying formula Φ . By Φ we indicate the formula stemming from $\Phi(?_{x_1}\{\mathcal{A}_{x_1}\}, \dots, ?_{x_n}\{\mathcal{A}_{x_n}\})$ where every placeholder $?_{x_i}$ is assigned to a symbol in \mathcal{A}_{x_i} . Take, for example, the temporal structure \mathcal{L} associated to log $L = [\langle a, b, d, c \rangle, \langle a, b, a, c \rangle]$ and the query $\square(?_x\{a, c\} \rightarrow \diamond(?_y\{b, d\}))$. Starting from this query, we need to evaluate the LTL_f rules in set $F = \{\square(a \rightarrow \diamond b), \square(a \rightarrow \diamond d), \square(c \rightarrow \diamond b), \square(c \rightarrow \diamond d)\}$ over \mathcal{L} .

In the first step of our approach, we generate a set of LTL_f formulas by replacing the placeholders in the input query with all the possible combination of events specified with the query (possibly all the events in the input log). Each LTL_f formula is then passed to *evalForm* procedure, together with \mathcal{L} , the set of states of \mathcal{L} , namely S , and the alphabet \mathcal{A} .

The first call to procedure *evalForm* starts a sequence of recursive calls, aiming at decomposing query Φ into subformulas, thus building an evaluation tree. This tree is meant to unfold the (possibly nested) subformulas up to atomic propositions (the leaves). All parent nodes are therefore LTL_f or propositional operators. The number of child nodes depend on the arity of the operator (e.g., 1 for \neg and X , 2 for \wedge and U). For example, from $\square(a \rightarrow \diamond b)$ (every occurrence of a is followed by at least one occurrence of b) and from $(\neg b U a) \vee \square \neg b$ (every occurrence of b is preceded by at least one occurrence of a) the evaluation trees in Fig. 2 are generated. Once the recursive call returns the set of states in which the subformula holds true, these states are treated by the calling procedure, back to the root of the evaluation tree.

For evaluating an atomic proposition l (i.e., a leaf in the evaluation tree) in a set of states \hat{S} , we use the procedure *evalAtom* (Algorithm 2). In particular, if l is \top , the procedure returns all the states of \hat{S} . If l is \perp , the empty set is returned. If l is an atomic proposition corresponding to the occurrence of an event, the procedure returns all the states of \hat{S} in which such event occurs.

For evaluating an LTL_f formula Φ in a set of states \hat{S} , we use the procedure *evalForm* (Algorithm 3). If Φ is the negation of a child expression φ , a recursive invocation of *evalForm* is used to retrieve all the states in \hat{S} in which φ holds. Then, the procedure returns the complement of \hat{S} with respect to this set. If Φ is the disjunction of two expressions φ and ψ , the procedure *evalForm* identifies the sets containing all

the states in \hat{S} in which φ and ψ hold respectively. The procedure returns the union of these two sets. Similarly, if Φ is the conjunction of φ and ψ , the sets containing all the states in \hat{S} in which φ and ψ hold are built and the procedure returns the intersection of these sets. The implication and the equivalence of two expressions are determined as combination of negation, disjunction and conjunction considering that $\varphi \rightarrow \psi$ is equivalent to $\neg(\varphi) \vee \psi$ and that $\varphi \leftrightarrow \psi$ is equivalent to $(\neg(\varphi) \vee \psi) \wedge (\neg(\psi) \vee \varphi)$.

If Φ consists of the *next* operator applied to a child expression φ , a recursive invocation of *evalForm* is used to retrieve all the states in $\hat{S} \triangleright$ (the set of the successors of \hat{S}) in which φ holds. The output set \hat{S}_\circ is the set of all the predecessors of these states. If Φ is the *future* operator applied to a child expression φ , a recursive invocation of *evalForm* is used to retrieve all the states in \hat{S} in which φ holds. The output set \hat{S}_\circ is initialized with these states. Iteratively all the predecessors of the states in \hat{S}_\circ are added to \hat{S}_\circ up to the initial states. If Φ is the *globally* operator applied to a child expression φ , the algorithm identifies the final states in which φ is satisfied. The output set \hat{S}_\square is initialized with these states. Iteratively all the predecessors of the states in \hat{S}_\square in which φ holds are added to \hat{S}_\square up to the initial states. Finally, if Φ is the *until* operator applied to two expressions φ and ψ , the procedure identifies the states in \hat{S} in which ψ is satisfied. The output set \hat{S}_U is initialized with these states. Iteratively all the predecessors of the states in \hat{S}_U in which φ holds are added to \hat{S}_U up to the initial states.

Consider, for example, the evaluation tree in Fig. 2a and the log temporal structure \mathcal{L} stemmed from $L = [\langle a, b, d, c \rangle, \langle a, b, a, c \rangle]$ (Fig. 1c). To check if formula $\square(a \rightarrow \diamond b)$ holds in L , the algorithms just described are invoked recursively from the root to the leaves of the evaluation tree of $\square(a \rightarrow \diamond b)$. The evaluation of the atomic proposition corresponding to the occurrence of a produces as result $\hat{S}_a = \{s_{1,1}, s_{2,1}, s_{2,3}\}$. The same algorithm applied to the node of the tree corresponding to activity b produces as result $\hat{S}_b = \{s_{1,2}, s_{2,2}\}$. Then, starting from \hat{S}_b , procedure *evalForm* produces $\hat{S}_{\diamond b} = \{s_{1,1}, s_{1,2}, s_{2,1}, s_{2,2}\}$. Starting from \hat{S}_a and $\hat{S}_{\diamond b}$, *evalForm* produces $\hat{S}_{a \rightarrow \diamond b} = \{s_{1,1}, s_{1,2}, s_{1,3}, s_f, s_{2,1}, s_{2,2}\}$. The final outcome of the recursive invocation of *evalForm* is $\hat{S}_{\square(a \rightarrow \diamond b)} = \{s_{1,1}, s_{1,2}, s_{1,3}, s_f\}$. Since $S_{\square(a \rightarrow \diamond b)}$ contains the initial state $s_{1,1}$, mapped to the first element of \mathbf{t}_1 , $\mathbf{t}_1(1)$, we can conclude that $\square(a \rightarrow \diamond b)$ holds in \mathbf{t}_1 , i.e., \mathbf{t}_1 is a *witness* for $\square(a \rightarrow \diamond b)$. In the same way, $\square(a \rightarrow \diamond b)$ does not hold in $s_{2,1}$ (mapped to the first element of \mathbf{t}_2), i.e., \mathbf{t}_2 is a *counterexample* for $\square(a \rightarrow \diamond b)$.

Algorithm 2. *evalAtom*($\mathcal{L}, l, \hat{S}, \mathcal{A}$), evaluating atomic propositions l in states \hat{S} of temporal structure \mathcal{T} defined over alphabet \mathcal{A}

Input: $\mathcal{L} = \langle S, \triangleright, S_1, S_f, L, W \rangle$, atomic proposition l , set of states $\hat{S} \subseteq S$, alphabet \mathcal{A}

Output: $\hat{S}_l \subseteq \hat{S}$ where l is evaluated as true

- 1 **if** $l = \top$ **then return** \hat{S}
 - 2 **if** $l = \perp$ **then return** \emptyset
 - 3 **if** $l = a$ with $a \in \mathcal{A}$ **then**
 - 4 **return** $\{s \in \hat{S} \mid L(s, a) \mapsto \top\}$
-

Algorithm 3. $evalForm(\mathcal{L}, \Phi, \hat{S}, \mathcal{A})$, evaluating LTL_f formula Φ in states \hat{S} of temporal structure \mathcal{T} defined over alphabet \mathcal{A}

```

Input: Temporal Structure  $\mathcal{L} = \langle S, \triangleright, S_1, S_f, L, W \rangle$ , LTLf formula  $\Phi$ , set of states  $\hat{S} \subseteq S$ , alphabet  $\mathcal{A}$ 
Output:  $\hat{S}_\Phi \subseteq \hat{S}$  where  $\Phi$  holds true

1 if  $\Phi \equiv \top$  or  $\Phi \equiv \perp$  or  $\Phi \equiv a$ , with  $a \in \mathcal{A}$  then
2   return  $evalAtom(\mathcal{T}, \Phi, \hat{S}, \mathcal{A})$ 
3 else if  $\Phi \equiv \neg\varphi$ , with  $\varphi$  LTLf formula over  $\mathcal{A}$  then
4   return  $\hat{S} \setminus evalForm(\mathcal{T}, \varphi, \hat{S}, \mathcal{A})$ 
5 else if  $\Phi \equiv \varphi \vee \psi$ , with  $\varphi, \psi$  LTLf formulas over  $\mathcal{A}$  then
6   return  $evalForm(\mathcal{T}, \varphi, \hat{S}, \mathcal{A}) \cup evalForm(\mathcal{T}, \psi, \hat{S}, \mathcal{A})$ 
7 else if  $\Phi \equiv \varphi \wedge \psi$ , with  $\varphi, \psi$  LTLf formulas over  $\mathcal{A}$  then
8   return  $evalForm(\mathcal{T}, \varphi, \hat{S}, \mathcal{A}) \cap evalForm(\mathcal{T}, \psi, \hat{S}, \mathcal{A})$ 
9 else if  $\Phi \equiv \varphi \rightarrow \psi$ , with  $\varphi, \psi$  LTLf formulas over  $\mathcal{A}$  then
10  return  $evalForm(\mathcal{T}, \neg\varphi \vee \psi, \hat{S}, \mathcal{A})$ 
11 else if  $\Phi \equiv \varphi \leftrightarrow \psi$ , with  $\varphi, \psi$  LTLf formulas over  $\mathcal{A}$  then
12  return  $evalForm(\mathcal{T}, \varphi \rightarrow \psi, \hat{S}, \mathcal{A}) \cap evalForm(\mathcal{T}, \psi \rightarrow \varphi, \hat{S}, \mathcal{A})$ 
13 else if  $\Phi \equiv \circ\varphi$ , with  $\varphi$  LTLf formula over  $\mathcal{A}$  then
14    $\hat{S}_\circ^\triangleright : evalForm(\mathcal{T}, \varphi, \hat{S}\triangleright, \mathcal{A})$ 
15    $\hat{S}_\circ : \triangleleft\hat{S}_\circ^\triangleright$ 
16   return  $\hat{S}_\circ$ 
17 else if  $\Phi \equiv \Diamond\varphi$ , with  $\varphi$  LTLf formula over  $\mathcal{A}$  then
18    $\hat{S}_\diamond : evalForm(\mathcal{T}, \varphi, \hat{S}, \mathcal{A})$ 
19    $\hat{S}_\diamond^\triangleleft : \triangleleft\hat{S}_\diamond$ 
20   while  $|\hat{S}_\diamond^\triangleleft| > 0$  do
21      $\hat{S}_\diamond : \hat{S}_\diamond \cup \hat{S}_\diamond^\triangleleft$ 
22      $\hat{S}_\diamond^\triangleleft : \triangleleft\hat{S}_\diamond^\triangleleft$ 
23   return  $\hat{S}_\diamond$ 
24 else if  $\Phi \equiv \Box\varphi$ , with  $\varphi$  LTLf formula over  $\mathcal{A}$  then
25    $\hat{S}_\square : evalForm(\mathcal{T}, \varphi, s_f, \mathcal{A})$ 
26    $\hat{S}_\square^\triangleleft : evalForm(\mathcal{T}, \varphi, \triangleleft\hat{S}_\square, \mathcal{A})$ 
27   while  $|\hat{S}_\square^\triangleleft| > 0$  do
28      $\hat{S}_\square : \hat{S}_\square \cup \hat{S}_\square^\triangleleft$ 
29      $\hat{S}_\square^\triangleleft : evalForm(\mathcal{T}, \varphi, \triangleleft\hat{S}_\square^\triangleleft, \mathcal{A})$ 
30   return  $\hat{S}_\square$ 
31 else if  $\Phi \equiv \varphi U \psi$ , with  $\varphi, \psi$  LTLf formulas over  $\mathcal{A}$  then
32    $\hat{S}_U : evalForm(\mathcal{T}, \psi, \hat{S}, \mathcal{A})$ 
33    $\hat{S}_U^\triangleleft : evalForm(\mathcal{T}, \varphi, \triangleleft\hat{S}_U, \mathcal{A})$ 
34   while  $|\hat{S}_U^\triangleleft| > 0$  do
35      $\hat{S}_U : \hat{S}_U \cup \hat{S}_U^\triangleleft$ 
36      $\hat{S}_U^\triangleleft : evalForm(\mathcal{T}, \varphi, \triangleleft\hat{S}_U^\triangleleft, \mathcal{A})$ 
37   return  $\hat{S}_U$ 

```

Given $\langle \hat{S}_\Phi, \Phi \rangle \in \mathcal{B}_\Phi^{\hat{S}}$, where $\mathcal{B}_\Phi^{\hat{S}}$ is the result of the call of $eval(\mathcal{L}, \Phi(?_{x_1}\{\mathcal{A}_{x_1}\}, \dots, ?_{x_n}\{\mathcal{A}_{x_n}\}), \mathcal{A})$, for a formula Φ , we compute:

- the number of witnesses, as $\sum_{\hat{s} \in (\hat{S}_\Phi \cap \mathcal{L}.S_i)} W(\hat{s})$, and
- the number of counterexamples, as $\sum_{\bar{s} \in (\mathcal{L}.S_i \setminus \hat{S}_\Phi)} W(\bar{s})$.

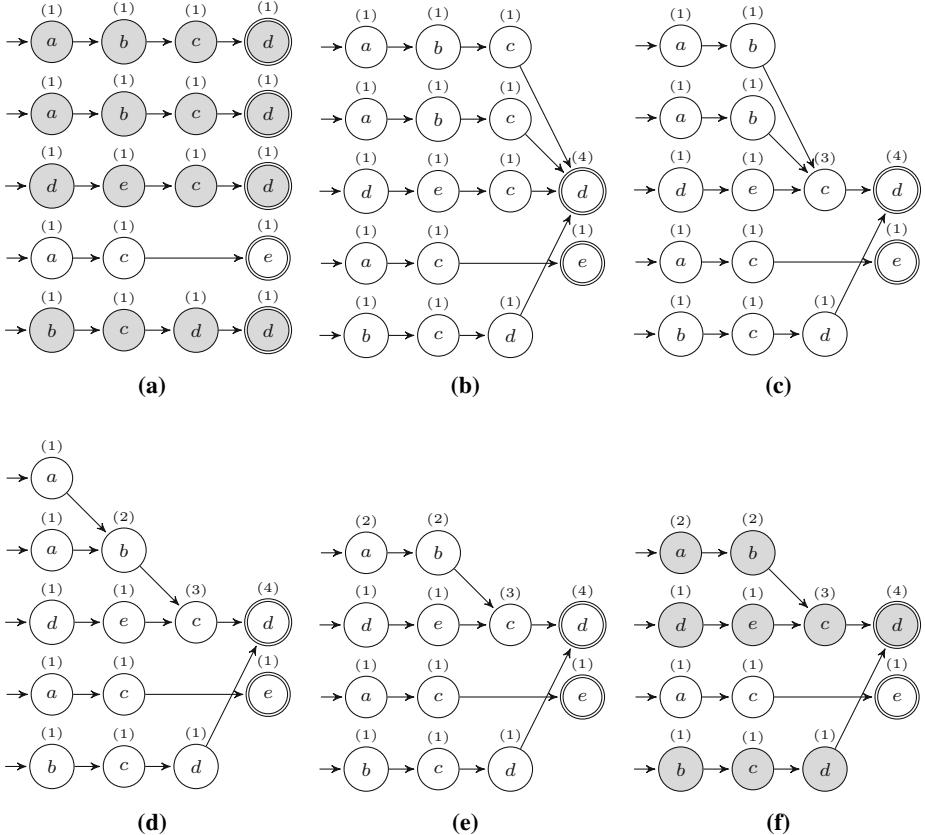


Fig. 3. Transformation into log temporal structure \mathcal{L} of log $L = \{\langle a, b, c, d \rangle, \langle a, b, c, d \rangle, \langle d, e, c, d \rangle, \langle a, c, e \rangle, \langle b, c, d, d \rangle\}$, defined over log alphabet $\mathcal{A} = \{a, b, c, d, e\}$. Fig. 3a and Fig. 3f highlight the states satisfying $\square(a \rightarrow \Diamond b)$.

The number of witnesses corresponds to the sum of weights of initial states of \mathcal{L} which are contained in the sets of states that satisfy Φ , \hat{S}_Φ . Conversely, the number of counterexamples corresponds to the sum of weights of initial states of \mathcal{L} which are not in \hat{S}_Φ . Such calculation directly derives from Corollary 1.

4 Experimentation

We implemented the algorithms illustrated in the previous sections as a software encoded in C.² All tests have been executed on a machine running Windows 7 operating system on an Intel Core i7 CPU with 8GB of main memory.

² Source and executable files are available at <https://github.com/r2im/pickaxe>.

4.1 Benchmarks

In order to evaluate the performance of the algorithm, we have run a series of experiments aiming at showing evidence of the effect of input parameters on execution time. Logs have been artificially generated by the log generator module of MINERful [14,13], a declarative process mining tool. Traces in artificial logs were created by distributing symbols in the input log alphabet uniformly at random. The results of such tests, depicted in Fig. 4, show the trend of the computation time w.r.t. the change in the input parameters: size of the alphabet, number of traces, and length of traces. By default, parameters regarding the log and the log alphabet were initialized as follows: the log consisted of 100 traces ($|L| = 100$), each being a sequence of 10 events ($|\mathbf{t}| = 10$), drawn from a log alphabet comprising 10 activities ($|\mathcal{A}| = 10$). The queries taken into account were those corresponding to the class of 18 constraint templates defined by the declarative process modeling language Declare [9] (see Sections 1 and 5) listed in [22,14], including, e.g., $\square(?_x \rightarrow \diamond?_y)$ and $(\neg?_y U ?_x) \vee \square \neg?_y$. Placeholders have been set free to be assigned to any symbol in the log alphabet. The reported computation time is the sum of the computation times for each query. Fig. 4a, Fig. 4b and Fig. 4c show the trend of computation time when altering, resp.: (i) the size of the alphabet, from 5 to 50; (ii) the number of traces, from 400 to 4000; (iii) the length of traces, from 5 to 50. The first and the second graph show a linear trend, whereas the third one draws an exponential curve. The reason resides in the construction of the log temporal structure: its states tend to linearly increase with the length of traces. Therefore, the search space grows exponentially. The folding of traces into the polytree of the log temporal structure seems to limit the state space increase when the number of traces or the number of activities grow.

Table 1 summarizes the changes in the execution time when query parameters change, namely (i) the LTL_f operator (either temporal or propositional), and (ii) the number of placeholders. Results are depicted, respectively, in Table 2a and Table 2b. In both cases, the log is created using the following parameters: $|L| = 4000$, $|\mathbf{t}| = 10$, $|\mathcal{A}| = 10$. Remarkably, Table 2a shows that the possibility to visit the temporal structure only once per formula makes the evaluation of $\circ?_x$ the fastest to be performed (see Algorithm 3). Conversely, $\diamond?_x$ turns out to be the most expensive in this regard. Table 2b highlights that the increase of placeholders entails an increase in the execution time approx. by an order of 10. This is due to the invocation of the *evalForm* procedure for every combination of symbols in the alphabet, assigned to placeholders (see Algorithm 1).

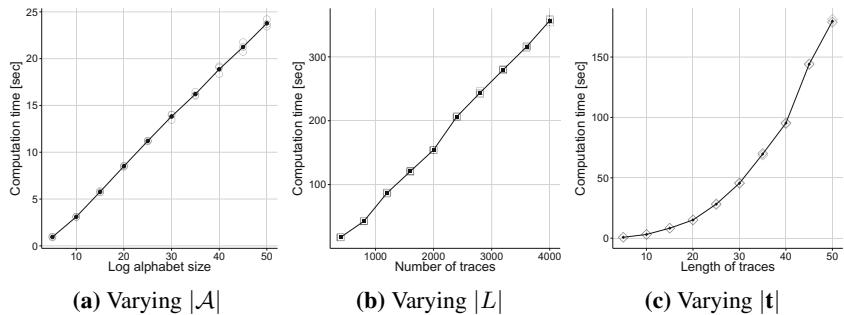
4.2 Case Study

We have conducted a case study by using the BPI challenge 2011 [1] event log. This log pertains to a healthcare process and, in particular, contains the executions of a process related to the treatment of patients diagnosed with cancer in a large Dutch academic hospital. The whole event log contains 1,143 cases and 150,291 events distributed across 623 event classes (activities). Each case refers to the treatment of a different patient. The event log contains domain specific attributes that are both case attributes and event

Query	Time [msec]	Query	Time [msec]	Query	Time [msec]
? _x	78	¬(? _x)	109	□(? _x → ◊(? _{y0}))	37 628
? _x ∨ ? _y	592	? _x ∧ ? _y	94	□(? _x → ◊(? _{y0} ∨ ? _{y1}))	281 362
? _x → ? _y	249	? _x ↔ ? _y	686	□(? _x → ◊(? _{y0} ∨ ? _{y1} ∨ ? _{y2}))	2 027 832
◊? _x	4 930	□ ? _x	422		
o? _x	93	? _x U ? _y	1 357		

(a) Varying query template

(b) Varying number of placeholders

Table 1. Performance evaluation w.r.t. query properties**Fig. 4.** Performance evaluation w.r.t. log properties and log alphabet.

attributes in addition to the standard XES attributes.³ For example, *Age*, *Diagnosis*, and *Treatment code* are case attributes and *Activity code*, *Number of executions*, *Specialism code*, and *Group* are event attributes.

To investigate the behavior of the process as recorded in the log, we have used the following LTL_f queries (in which the original Dutch language is not translated):

1. □((aannname laboratoriumonderzoek) → ◊?_x{ca-19.9 tumormarker, cea - tumormarker mbv meia, ca-125 mbv meia});
2. ◊?_x{ureum, albumine, creatinine} ∧ ◊?_y{calcium, glucose, natrium vlamfotometrisch, kalium potentimetrisch};
3. (¬?_x{vervolgconsult poliklinisch, 1e consult poliklinisch, telefonisch consult} U (aannname laboratoriumonderzoek)) ∨ □(¬?_x{vervolgconsult poliklinisch, 1e consult poliklinisch, telefonisch consult});
4. ¬(◊(aannname laboratoriumonderzoek) ∧ ◊?_x{ct abdomen, ct thorax, ct bovenbuik, ct hersenen, thorax}).

In Table 2, we show the solutions of these queries. For each solution, we show the number of cases in which the solution holds (witnesses) and the number of

³ XES (eXtensible Event Stream) is an XML-based standard for event logs proposed by the IEEE Task Force on Process Mining (www.xes-standard.org).

Ws	Cs	Solution
344	799	$\square((aanname laboratoriumonderzoek) \rightarrow \diamond(ca-19.9 tumormarker))$
431	712	$\square((aanname laboratoriumonderzoek) \rightarrow \diamond(cea - tumormarker mbv meia))$
517	626	$\square((aanname laboratoriumonderzoek) \rightarrow \diamond(ca-125 mbv meia))$

(a) Solutions for query $\square((aanname laboratoriumonderzoek) \rightarrow \diamond?_x\{ca-19.9 tumormarker, cea - tumormarker mbv meia, ca-125 mbv meia\})$

Ws	Cs	Solution	Ws	Cs	Solution
627	516	$\diamond(ureum) \wedge \diamond(calcium)$	627	516	$\diamond(albumine) \wedge \diamond(calcium)$
640	503	$\diamond(creatinine) \wedge \diamond(glucose)$	627	516	$\diamond(ureum) \wedge \diamond(glucose)$
623	520	$\diamond(albumine) \wedge \diamond(glucose)$	693	450	$\diamond(creatinine) \wedge \diamond(natrium vlamfotometrisch)$
663	480	$\diamond(ureum) \wedge \diamond(natrium vlamfotometrisch)$	640	503	$\diamond(albumine) \wedge \diamond(natrium vlamfotometrisch)$
697	446	$\diamond(creatinine) \wedge \diamond(kalium potentimetrisch)$	665	478	$\diamond(ureum) \wedge \diamond(kalium potentimetrisch)$
642	501	$\diamond(albumine) \wedge \diamond(kalium potentimetrisch)$	634	509	$\diamond(creatinine) \wedge \diamond(calcium)$

(b) Solutions for query $\diamond?_x\{ureum, albumine, creatinine\} \wedge \diamond?_y\{calcium, glucose, natrium vlamfotometrisch, kalium potentimetrisch\}$

Ws	Cs	Solution
797	346	$\neg(vervolgconsult poliklinisch) U(aanname laboratoriumonderzoek) \vee \square(\neg(vervolgconsult poliklinisch))$
1046	97	$\neg(Ie consult poliklinisch) U(aanname laboratoriumonderzoek) \vee \square(\neg(Ie consult poliklinisch))$
1041	102	$\neg(telefonisch consult) U(aanname laboratoriumonderzoek) \vee \square(\neg(telefonisch consult))$

(c) Solutions for query $(\neg?_x\{vervolgconsult poliklinisch, Ie consult poliklinisch, telefonisch consult\} U(aanname laboratoriumonderzoek)) \vee \square(\neg?_x\{vervolgconsult poliklinisch, Ie consult poliklinisch, telefonisch consult\})$

Ws	Cs	Solution
811	332	$\neg(\diamond(aanname laboratoriumonderzoek) \wedge \diamond(ct abdomen))$
1012	131	$\neg(\diamond(aanname laboratoriumonderzoek) \wedge \diamond(ct thorax))$
1127	16	$\neg(\diamond(aanname laboratoriumonderzoek) \wedge \diamond(ct bovenbuik))$
1122	21	$\neg(\diamond(aanname laboratoriumonderzoek) \wedge \diamond(ct hersenen))$
637	506	$\neg(\diamond(aanname laboratoriumonderzoek) \wedge \diamond(thorax))$

(d) Solutions for query $\neg(\diamond(aanname laboratoriumonderzoek) \wedge \diamond?_x\{ct abdomen, ct thorax, ct bovenbuik, ct hersenen, thorax\})$

Table 2. Query solutions for the proposed case study (in column headers, “Ws” stands for “Witnesses” and “Cs” for “Counterexamples”)

cases in which the solution is not valid (counterexamples). For all the queries in Table 2a, there are more counterexamples than witnesses. For example, for $\square((aanname laboratoriumonderzoek) \rightarrow \diamond(ca-19.9 tumormarker))$, there are 799 counterexamples and only 344 witnesses meaning that in 821 cases out of 1,143 at least one occurrence of *aanname laboratoriumonderzoek* (*assumption laboratory*) is not eventually followed by *ca-19.9 tumormarker* in the same case. For queries in Ta-

bles 2b, 2c, and 2d the number of witnesses is higher than the number of counterexamples. The number of witnesses and the number of counterexamples for solutions in Table 2b are more balanced. For example, for $\Diamond(\text{albumine}) \wedge \Diamond(\text{glucose})$, there are 623 witnesses and 520 counterexamples. This means that in 623 cases *albumine* and *glucose* coexist in the same case and in 520 cases they do not. For solutions in Tables 2c and 2d, there are more witnesses supporting the validity of the rules. From Table 2c, we can derive that very often a medical consultation is preceded by laboratory tests (*assumption laboratory*). For example, in 1,041 cases out of 1,143, *telefonisch consult* (*telephonic consultation*) is preceded by *aanname laboratoriumonderzoek* (*assumption laboratory*). In 1,046 cases out of 1,143, *1e consult poliklinisch* (*First outpatient visit*) is preceded by *aanname laboratoriumonderzoek*. From Table 2d, we can conclude that very rarely laboratory tests coexists with computed tomography (ct) tests. For example, for $\neg(\Diamond(\text{aanname laboratoriumonderzoek}) \wedge \Diamond(\text{ct bovenbuik}))$, there are 1,127 witnesses and only 16 counterexamples meaning that in 1,127 cases out of 1,143 *aanname laboratoriumonderzoek* and *ct bovenbuik* (*ct upper abdomen*) do not coexist in the same case.

5 Related Work

Several approaches in the literature focus on the discovery of declarative process models [6,14,18,20,21,22,19]. In particular, the technique proposed in [12,14] is based on a two-step approach. First, the input event log is parsed to generate a knowledge base containing information useful to discover a Declare model. Then, in a second phase, the knowledge base is queried to find the set of Declare constraints that hold on the input log. The work proposed in [20] is based on an Apriori algorithm for association rule mining. In [19], the authors propose an algorithm for the online discovery of Declare rules. These works propose “ad hoc” algorithms for the discovery of a limited class of business rules. In contrast, our method can be used to discover any type of LTL_f -based constraint.

The approaches proposed in [6,18] are more general and allow for the specification of rules that go beyond the traditional Declare templates. However, these approaches can be hardly used in real-life settings since they are based on supervised learning techniques and, therefore, they require negative examples. In [21], a first-order variant of LTL is used to specify a set of data-aware patterns. Although this approach supports constraints involving data conditions, it can only be applied to discover the (limited) set of standard Declare rules.

Several approaches exist for temporal logic query checking. For example, Chan propose an approach that can cope with single placeholders in queries only appearing once as positive literals [3]. To overcome these limitations, Bruns and Godefroid propose a theoretical approach based on Extended Alternating Automata (EEA) [2,17]. Gurfinkel et al. describe their query checking tool named TLQSolver, capable of dealing with multiple placeholders, both appearing as negated or positive, and occurring several times in the query [15]. It is also proved that query checking is an instance of multi-valued model checking [5], i.e., a generalization of a classical model checking problem, where the model remains such that both transition relations and atomic propositions are two-valued, but lattice values are allowed to appear as constants in temporal logic formulas.

Indeed, their solution is based on their existing multi-valued model checker, \mathcal{X} Check [4]. These approaches have been implemented to deal with CTL query checking. Our approach is inspired by these works, but adapts them efficiently to LTL_f , in order to become applicable also in a real-life context.

6 Conclusion

In this paper, we have introduced an approach in the middle between process discovery and conformance checking, based on temporal logic query checking. In particular, our proposed technique produces as outcome a set of LTL-based business rules as well as diagnostics in terms of witnesses (traces of the input log in which each rule is satisfied) and counterexamples (traces of the input log in which each rule is violated). There are several directions for future work. It is possible to optimize the checking algorithms for improving performances, by further exploiting the characteristics of the contexts in which our technique is applied, such as the finiteness of the traces in an event log. More sophisticated techniques can be used to choose the best replacements for placeholders in a query. For example, the Apriori algorithm proposed in [20] can be used to automatically choose the set of activities to be assigned as replacements based on the frequency of their co-occurrence in a case.

References

1. 3TU Data Center. BPI Challenge, Event Log (2011), doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
2. Bruns, G., Godefroid, P.: Temporal logic query checking. In: LICS, pp. 409–417. IEEE Computer Society (2001)
3. Chan, W.: Temporal-logic queries. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 450–463. Springer, Heidelberg (2000)
4. Chechik, M., Devereux, B., Easterbrook, S.M., Gurinkel, A.: Multi-valued symbolic model-checking. ACM Trans. Softw. Eng. Methodol. 12(4), 371–408 (2003)
5. Chechik, M., Easterbrook, S.M., Petrovykh, V.: Model-checking over multi-valued logics. In: Oliveira, J.N., Zave, P. (eds.) FME 2001. LNCS, vol. 2021, pp. 72–98. Springer, Heidelberg (2001)
6. Chesiari, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Exploiting inductive logic programming techniques for declarative process mining. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 278–295. Springer, Heidelberg (2009)
7. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 8(2), 244–263 (1986)
8. Clarke, E.M., Grumberg, O., Peled, D.: Model checking. MIT Press (2001)
9. De Giacomo, G., De Masellis, R., Montali, M.: Reasoning on LTL on finite traces: Insensitivity to infiniteness. In: AAAI (2014)
10. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI (2013)
11. Deutch, D., Milo, T.: A structural/temporal query language for business processes. J. Comput. Sci. 78(2), 583–609 (2012)

12. Di Cicco, C., Mecella, M.: Mining constraints for artful processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS 2012. LNIP, vol. 117, pp. 11–23. Springer, Heidelberg (2012)
13. Di Cicco, C., Mecella, M.: Studies on the discovery of declarative control flows from error-prone data. In: SIMPDA, pp. 31–45 (2013)
14. Di Cicco, C., Mecella, M.: A two-step fast algorithm for the automated discovery of declarative workflows. In: CIDM, pp. 135–142. IEEE (2013)
15. Gurfinkel, A., Chechik, M., Devereux, B.: Temporal logic query checking: A tool for model exploration. IEEE TSE 29(10), 898–914 (2003)
16. Hildebrandt, T.T., Mukkamala, R.R., Slaats, T., Zanitti, F.: Contracts for cross-organizational workflows as timed dynamic condition response graphs. J. Log. Algebr. Program. 82(5–7), 164–185 (2013)
17. Kupferman, O., Vardi, M.Y., Wolper, P.: An automata-theoretic approach to branching-time model checking. J. ACM 47(2), 312–360 (2000)
18. Lamma, E., Mello, P., Riguzzi, F., Storari, S.: Applying inductive logic programming to process mining. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS (LNAI), vol. 4894, pp. 132–146. Springer, Heidelberg (2008)
19. Maggi, F.M., Burattin, A., Cimitile, M., Sperduti, A.: Online process discovery to detect concept drifts in LTL-based declarative process models. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) ODBASE 2013. LNCS, vol. 8185, pp. 94–111. Springer, Heidelberg (2013)
20. Maggi, F.M., Bose, R.P.J.C., van der Aalst, W.M.P.: Efficient discovery of understandable declarative process models from event logs. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 270–285. Springer, Heidelberg (2012)
21. Maggi, F.M., Dumas, M., García-Bañuelos, L., Montali, M.: Discovering data-aware declarative process models from event logs. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 81–96. Springer, Heidelberg (2013)
22. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: CIDM, pp. 192–199. IEEE (2011)
23. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: Declare: Full support for loosely-structured processes. In: EDOC, pp. 287–300. IEEE (2007)
24. Pnueli, A.: The temporal logic of programs. In: FSTTCS, pp. 46–57. IEEE (1977)
25. 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)
26. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Software Eng. 37(3), 410–429 (2011)

Approach and Refinement Strategies for Flexible Choreography Enactment

Andreas Weiß, Santiago Gómez Sáez, Michael Hahn, and Dimka Karastoyanova

Institute of Architecture of Application Systems (IAAS)
University of Stuttgart, Stuttgart, Germany
`{andreas.weiss, santiago.gomez-saez, michael.hahn,
dimka.karastoyanova}@iaas.uni-stuttgart.de`

Abstract. Collaborative, Dynamic & Complex (CDC) systems such as adaptive pervasive systems, eScience applications, and complex business systems inherently require modeling and run time flexibility. Since domain problems in CDC systems are expressed as service choreographies and enacted by service orchestrations, we propose an approach introducing placeholder modeling constructs usable both on the level of choreographies and orchestrations, and a classification of strategies for their refinement to executable workflows. These abstract modeling constructs allow deferring the modeling decisions to later points in the life cycle of choreographies. This supports run time scenarios such as incorporating new participants into a choreography after its enactment has started or enhancing the process logic of some of the participants. We provide a prototypical implementation of the approach and evaluate it by means of a case study.

Keywords: Process Flexibility, Choreography Flexibility, Refinement Strategies, Late Modeling, Late Selection, Process Fragments.

1 Introduction

In our research in Collaborative, Dynamic & Complex systems (CDC) [3] we aim at supporting several application domains exhibiting overlapping requirements, but so far diverging solutions, with unified concepts, techniques and tools. The domains we consider are scientific experiments and workflows, pervasive adaptive systems, service networks, configurable multi-tenant applications (see also Sec. 2.1). We have introduced a three-phase life cycle of CDC systems comprising Modeling, Provision, and Execution phases (cf. Sec. 3). To capture the overall communication behavior of applications in such systems, we use choreographies. Choreographies are a concept known from the business domain that enables independent organizations to collaborate and reach a common business goal. They provide a global view on the interconnection of independent, but collaborating organizations, which are denoted by choreography participants, communicating without a central coordinator [8]. The publicly visible communication interfaces of each choreography participant are enacted, i.e., implemented by services or orchestrations of services (workflows). In previous work, we have enabled the

enactment of choreographies modeling CDC systems through a mapping/transformation to abstract workflows (containing only the communication activities), and their manual refinement to executable service orchestrations for the above mentioned domains [3]. This implies that the CDC Modeling phase is divided into a choreography modeling, a transformation, and a workflow modeling phase. A central concept of CDC systems is flexibility. In our work and in numerous related works, concepts, methods and techniques for flexible workflows have been studied in detail [24], [25]. Generally, the flexibility aspects can be divided into means to arbitrarily change workflows during modeling or run time [19], [24] or means for changes in predefined regions (cf. Sec. 2.3). Flexibility in choreographies, however, while instrumental for CDC systems, has not yet been studied as much as workflow flexibility. The few existing works on arbitrary choreography adaptation, changes and change propagation such as [10], [21], [27] are only one aspect of choreography flexibility. Towards reaching the goal of more choreography related flexibility, we propose an approach in Sec. 3 that introduces abstract placeholders/constructs, namely abstract activities and abstract connectors, on the level of choreographies in order to defer choreography modeling decisions to a later point in time. That means, our approach enables changes in predefined regions considering also the choreography level and refinement of the regions in all CDC life cycle phases. We call this end-to-end flexibility. We identify a set of strategies to refine the abstract constructs and position them with respect to the life cycle of CDC systems. The need for placeholders and their refinement not only on the orchestration level but also on the level of choreographies is explicitly motivated using three application domains (Sec. 2.1) and an motivating example (Sec. 2.2) to derive a set of requirements. These requirements are compared to related work to show the research gap we aim to fill with our approach (Sec. 2.3). Furthermore, we present a prototypical implementation of our approach (Sec. 4), which includes the extension of the choreography language BPEL4Chor [9] and the workflow language BPEL [18], and evaluate the approach in a case study. The paper is concluded with a summary and outlook to future work (Sec. 5).

2 Motivation and Related Work

2.1 Motivating Scenarios

In our research we are working towards enabling IT support for scenarios from three different fields: scientific experiments, collective adaptive systems, and configurable multi-tenant applications. We have already shown that these areas impose overlapping requirements on the supporting applications systems [3] and we introduced the notion of CDC systems as a type of applications fulfilling these requirements. Since in this work we focus on the *dynamic* aspect of CDCs, here we will identify the requirements on CDC systems that, when fulfilled, will significantly improve their flexibility.

Scientific (multi-scale and multi-field) experiments: In our work in the scope of the Cluster of Excellence Simulation Technology (SimTech¹) we want to enable

¹ SimTech: <http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/>

scientists to model and execute multi-scale and multi-field experiments in a user friendly manner. Previously [24], concepts and a workflow management system for supporting the trial-and-error nature of scientific simulations have been developed. Workflows that are not completely specified can be started and altered during run time. These simulation workflows orchestrate scientific services dealing with one scientific field which operates on one scale. For scientists, however, it is extremely important to be able to couple scientific models representing distinct scientific fields, such as biology, chemistry, and physics, and operating on different length scales, such as micro, macro or continuum scale, just by combining the already available simulation applications in a more complex application that represents the multi-scale and multi-field experiment. For this, we model the overall experiment as a choreography which is enacted by simulation workflows. The support for the trial-and-error manner of interleaved modeling and execution of experiments and the cooperative nature of the interactions among scientists is also particularly important. When collaboratively modeling the multi-scale and multi-field experiment as a choreography, scientists may want to define with an abstract placeholder that certain simulation steps have to be conducted such as building a Finite Element Method (FEM) but the concrete specification is left open for the particular scientist with the expertise to refine the enacting simulation workflow. The abstract placeholder defined on the choreography level would provide some guidance for refinement on the workflow level. The refinement of the abstract placeholders may even be conducted after the experiment has been started. This would allow to take intermediate results into account when deciding on the next concrete steps in a choreography of scientific workflows. Furthermore, in order to hide technical details of the communication between different scientific fields and scales, facilities to define communication relationships in an abstract way and refine them later would be helpful for scientists not familiar with technical details of the execution environment.

Collective Adaptive Systems (CAS) comprise heterogeneous entities that collaborate towards achieving their own objectives, and the overall objective of the collective [2]. These entities can be either virtual or physical, and organizationally and geographically distributed. The interaction of such entities with the collective highly influences the behavior of the system, as individual entity objectives may conflict with the behavior or decisions of other entities. Furthermore, unexpected changes in the entities' environment may trigger an individual or collective behavioral change. For purposes of providing a system capable of supporting this behavioral definition, monitoring, and adaptation in the EU ALLOW Ensembles project², we define the underpinning concepts for modeling, execution, and adaptation of CAS entities and their interactions. In particular, we propose to model and manage entities as collections of cells encapsulating their functionalities; cells are realized by service orchestrations. Entities collaborate with each other to achieve their objectives in the context of ensembles, enacted as choreographies, describing the interactions among them [2]. CAS are based on fundamental concepts from the so-called Adaptable Pervasive Flows

² ALLOW Ensembles: <http://www.allow-ensembles.eu>

(APF) [7], [13]. Adaptability of CAS is required on the level of both cells and ensembles and hence has to be supported by means of flexible orchestrations and the interactions among them in choreographies. The cells' behavior can be partially or completely specified during the modeling phase. The partial specification of the cell behavior covers the partial definition of one or more of their tasks as abstract tasks, which must be refined by concrete tasks during run time.

Configurable Workflows is a research area that we investigate towards enabling multi-tenant applications, i.e., applications that are designed to maximize the resource sharing between multiple consumers. Configurable applications are capable to adapt their behavior or appearance without the necessity to change the underlying implementation [11], [14] to the needs of entities (e.g., tenants, tenant users). The *multi-tenant aware Service Composition Engine* (SCE^{MT}) introduced in [12] is one example of a SCE that can be shared by different entities. Furthermore, it enables users to customize predefined workflow models based on their needs by providing a set of configuration data without the necessity to create new or adapt existing workflow models. These configuration data are used in all instances of a workflow model which belong to a corresponding user. Currently, SCE^{MT} supports the registration of runtime data (e.g., variable or service endpoint values). On the one hand, this enables users to customize an existing model by overriding predefined values (e.g., constants specified in a workflow model). On the other hand, workflow modelers are able to define more generic and reusable models (process templates) which can be adapted by the users themselves through configuration. For example, any country specific values of a payment process such as tax rates can be dynamically replaced during run time by using registered model configurations related to the user who instantiates the workflow model. In order to provide a user the ability to customize parts of the control flow logic of a workflow model, a modeler should be able to insert placeholders into a workflow model. The placeholders can be refined with the registered logic of a user during a later point in the life cycle of the workflow. This provides a much more general, flexible and powerful solution for the configuration of workflow models because a user would be able to customize the control flow and therefore the behavior of a workflow model. Configurability is also required on the level of choreographies which would enable the configuration of the entire set of interconnected workflow models by a user.

2.2 Example and Requirement Identification

Fig. 1 shows a trip booking example from the CAS domain to illustrate the need for placeholder flexibility constructs beginning from the choreography level to the execution of the enacting workflows. The example forms a choreography consisting of three participants showing only their public interfaces: the passenger, the passenger management system, and a payment manager. The passenger participant initiates a conversation with the passenger management system exchanging trip and payment details. After specifying the payment details, the payment manager participant comes into play. Let's assume that the modeler of the choreography only wants to model a communication relationship between

the passenger management system and the payment manager and does not care about the technical details of the communication. That means for example that it is left unspecified by the modeler if the communication pattern is synchronous or asynchronous. Furthermore, the choreography modeler only knows that the payment manager participant will somehow conduct a payment depending on the payment preferences of the passenger. However, it is left unspecified how the payment will be processed and whether other participants will join the choreography to execute the payment. To realize this kind of flexibility that allows the specification of an abstract placeholder for the logic to invoke services or other participants during the Modeling phase of a CDC system but defers the decision about the concrete workflow logic to a later point in time (at latest to run time), two concepts are needed: (i) a placeholder for workflow logic and (ii) a logical connector between choreography participants that can be refined later.

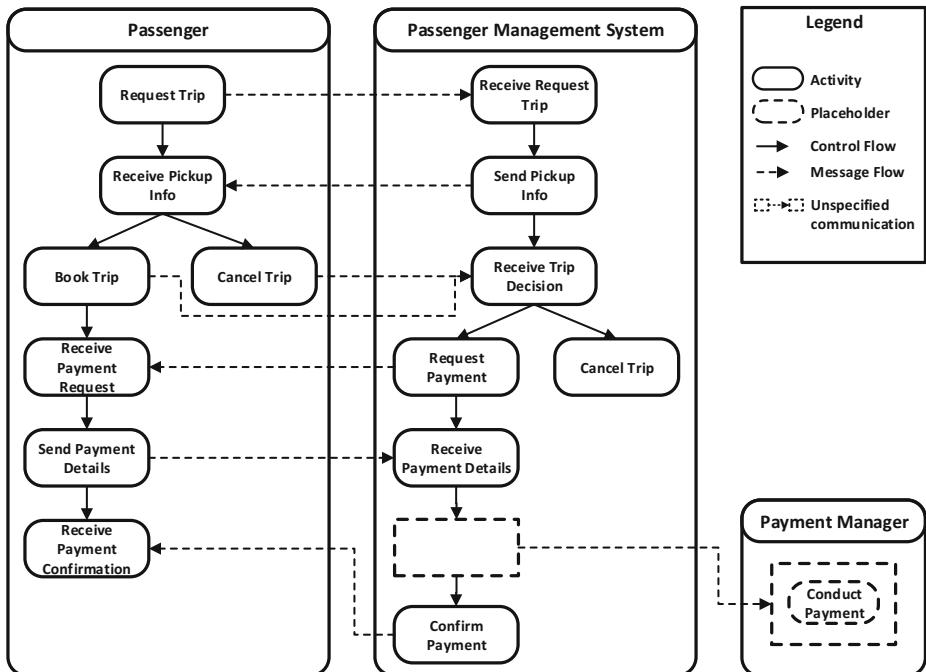


Fig. 1. Trip booking example from the Collective Adaptive System domain

Based on the three application domains in general and the motivating example in particular, we identify the following requirements for an approach to enable the discussed end-to-end flexibility features on both the level of choreographies and the enacting workflows.

R1. Choreography placeholders: Choreography modelers need capabilities for specifying placeholders in their models to defer decisions about concrete

orchestration steps. The refinement of the placeholders must be feasible in any CDC life cycle phase.

- R2. **Workflow placeholders:** Workflow modelers refining the generated non-executable workflows which enact a choreography must also have access to placeholder constructs to defer the decision about orchestration steps to the Provision or Execution phase.
- R3. **Same semantics:** The placeholder constructs should have the same semantics on both the choreography and workflow level and modelers should be able to use them consistently on both levels.
- R4. **Manual and automatic refinement:** The refinement of placeholders with process fragments must be possible both manually by human modelers and automatically through a model (choreography or workflow) fragment discovery operation.
- R5. **Execution of unrefined workflows:** Workflows having unrefined placeholders must still be executable, i.e., the workflows must be instantiable and executable until a placeholder is reached. Then either a manual or automatic refinement must take place.
- R6. **Phase annotation:** The placeholder must be annotated with information in which life cycle phase the refinement has to take place. The CDC system has to enforce the annotated information.
- R7. **Abstract definition of collaborations:** It must be possible for choreography modelers to specify the collaboration between participants in an abstract manner to defer the decision about the technical realization in terms of transport protocol and Message Exchange Pattern (MEP). This helps non-technical users of the system to avoid dealing with technical details.
- R8. **Discovery of participants:** The definition of abstract collaborations of choreography participants which are unknown at design time but dynamically discovered during run time must be possible. That means a placeholder indicates the communication with an not yet known participant.
- R9. **Generality:** The concept of the placeholders should be generic and not tailored to a specific application domain, but also support the idiosyncrasies of different domains.

2.3 Review of Existing Concepts

The central concept needed by all three presented domains is an abstract placeholder (on the level of choreographies and workflows) that can be refined with concrete workflow logic even after the Modeling life cycle phase. The workflow logic for refinement contains either single tasks, also called activities, or process fragments, which are a set of activities connected by control and data flow. Existing work uses the concept of predefined regions representing placeholders in workflow models that can be refined after the instantiation of the model. The refinement is denoted by *Late modeling of process fragments* in case the activities or process fragments are newly specified inside a placeholder during run time and *Late selection of process fragments* in case a set of activities or process fragments has been pre-modeled but the actual selection happens during run time

based on predefined rules or user decisions [25]. There are several approaches realizing these generic patterns. In [1], the notion of worklets, which are completely specified YAWL processes that refine a worklet enabled parent YAWL task at run time, is used for workflows in order to provide workflow run time flexibility. The selection of the most appropriate worklet for a worklet enabled task is based on rules considering context information. In [22], the concept of Pockets of Flexibility (PoF) is introduced. The PoF which contain a set of activities and the control flow between the activities are defined during run time, either (semi-)automatically or manually depending on previously executed activities. Ardissono et al. use a planning based approach for retrieving sub-processes implementing a so-called abstract activity [4], while in [6] from the field of pervasive flows planning based techniques are used to generate a sub-process. The domain of Adaptable Pervasive Flows (APF) [7], [13] also uses the concept of abstract activities which are refined depending on the context of the entities represented by a workflow.

All these approaches have in common that they concentrate on providing flexibility on the level of individual workflows. They fulfill the requirement R2 for having a workflow placeholder, R5 for being executable even with unrefined regions, R4 demanding manual or automatic refinement, and are mostly generic and not limited to a particular domain as stated in R9. However, our scenarios explicitly demand the concept of predefined regions on the level of choreographies in order to give guidance for workflow refinement, to incorporate further participants during later life cycle phases or defer the decision about communication patterns to later phases (R1, R3, R6, R7, R8). To the best of our knowledge, these requirements are not fulfilled by the existing concepts.

Regarding choreography flexibility in general, i.e., without the use of placeholders, several approaches exist. For example, in [21] the propagation of changes appearing in the private process of one choreography participant to the affected business partners is enabled without considering already running choreographed workflow instances. Formal methods are used to calculate if and what changes are necessary and if the new message interchange is deadlock free. In [27], an improved formal approach is introduced for change propagation from choreographies to orchestrations. In [10], a generic approach for propagation of local changes to directly as well as transitively dependent business partners is shown. A framework for collaborative choreography customization is presented in [16]. The so-called global (choreographies) and local views of each participant are expressed in the agent-oriented software development methodology Tropos. Participants agree on customization alternatives that best fulfill their business needs, i.e., their local view. Basically, such approaches enable the change of already existing choreography models and propagate changes to the enacting workflows and services. They are orthogonal to our requirement for abstract modeling constructs in order to enable end-to-end flexibility through late modeling and selection for choreographies and their enacting workflows. To the best of our knowledge, there are no available approaches satisfying this requirement. In Sec. 3 we propose a new approach to close this research gap.

3 End-to-End Flexibility and Refinement Strategies

Based on the requirements identified in Sec. 2.2, we present our approach for providing end-to-end flexibility throughout the Modeling, Provision, and Execution life cycle phases of CDC systems which are realized through choreographies and orchestrations. We propose the use of *abstract constructs* during the different life cycle phases of a CDC system. These abstract constructs are denoted in the scope of our work as abstract activities and abstract connectors.

3.1 Abstract Constructs

We define an *abstract activity* as a placeholder in a choreography or workflow model, which should be refined to concrete choreography or orchestration logic immediately before its execution at the latest. An abstract activity is connected to the preceding and succeeding activities by control and data flow. Definition 1 expresses formally the properties of an abstract activity.

Definition 1 (Abstract Activity). *An abstract activity α is represented by the tuple $\alpha = (n, \alpha, \omega, \tau, \epsilon, T, \sigma)$ where n is the name of α , α specifies the life cycle phase in which the refinement of α may first be conducted, ω specifies the life cycle phase in which the refinement of α has to be conducted at latest, τ is the plugin type of α , ϵ is the failure handling strategy for α , T denotes the type-specific content of α , and σ is the plugin selection function $\sigma : \tau \mapsto T$, which maps the plugin type τ to the type-specific content T . For α and ω the following applies: $\alpha, \omega \in \{\text{"choreography_modeling"}, \text{"workflow_modeling"}, \text{"provision"}, \text{"execution"}, \emptyset\}$. For the failure handling strategy ϵ applies: $\epsilon \in \{\text{"none"}, \text{"retry"}, \text{"manual"}\}$.*

An abstract activity α is a generic placeholder that can be typed through the plugin type attribute τ . This attribute allows the assignment of different implementations realizing different refinement behavior or simply indicates that the refinement has to be conducted manually. The plugin selection is formally defined by the plugin selection function σ mapping the plugin type of τ to the type-specific content T of the abstract activity (i.e., the plugin). The plugin selection function σ is important for providing the correct execution plugin in a workflow engine and the correct user interface plugin for modeling in a modeling tool. For example to take into account the context and objectives of the entities represented by the workflows for the refinement of an abstract activity, the value *APF_AdaptationManager* could be assigned for τ . This implies that the type-specific content T is populated with concepts from the Adaptable Pervasive Flows domain (APF) [7], [13]. For example, *preconditions* and *effects* attributes are introduced that specify the desired behavior of the logic the abstract activity represents without limiting it to predefined execution patterns. The preconditions specify in which state the context of the particular workflow has to be prior to the refinement of an abstract activity. The effects specify the desired state after the execution of the refined logic. A workflow engine implementation supporting the *APF_AdaptationManager* plugin expects the

specified type-specific attributes and contacts an external adaptation manager to retrieve a process fragment based on the context values. For other domains τ and T may have completely different values. The failure handling strategy ϵ of an abstract activity specifies how a supporting workflow engine should behave if $\sigma(\tau) = \emptyset$ in case an appropriate plugin is not installed in the engine. Possible strategies could be retrying the selection or asking a user for resolution.

The discovery of an appropriate fragment is the responsibility of the selected plugin, i.e., the type-specific content T of an abstract activity. The discovery is formally defined in Definition 2.

Definition 2 (Fragment Discovery). *The fragment discovery function $\delta : T \mapsto f$ is a function that maps the type-specific content T to a model fragment $f \in F$ where F is a set of process or choreography fragments. We denote the application of δ as the refinement of the abstract activity a .*

Each plugin implements the function δ offering different discovery methods/algorithms for example for manual or automatic selection. A further advantage of using a plugin concept is the possibility to specify different refinement/discovery methods inside an individual choreography or workflow model.

An *abstract connector* is a composite modeling element for choreographies that consists of two so-called *abstract containers* and a communication link. The abstract containers represent the source and the target of the communication link connecting the involved choreography participants. If one participant has to send messages to more than one participant, this is not modeled by an abstract connector but rather the choreography language itself has to provide means for expressing a set of participants whose number is not known at design time [9]. An abstract connector can be modeled between sending and receiving participants (or sets of them) in order to postpone the decision about the MEP to be used. Definition 3 describes the properties of an abstract connector formally.

Definition 3 (Abstract Connector). *An abstract connector c is represented by a tuple $c = (n, C_{\text{source}}, C_{\text{target}}, \lambda, \alpha, \omega, \tau, \epsilon, T)$ where n is the name of c , C_{source} is the abstract container that represents the source of c , C_{target} is the abstract container that represents the target of c , and $\lambda : C_{\text{source}} \mapsto C_{\text{target}}$ is the connector link of c . An abstract container C is itself a tuple $C = (n_C, \mathfrak{M})$ where n_C is the name and \mathfrak{M} is the content of the abstract container which may be empty or contain all elements for specifying control and data flow in the used choreography or orchestration language, as well as abstract activities and nested abstract containers. The remaining elements α , ω , τ , ϵ , and T have the same semantics as defined in Definition 1.*

The abstract containers C_{source} and C_{target} define the region in which the message exchange between two choreography participants has to take place. The content \mathfrak{M} of an abstract container C may be empty and thus only expressing the communication relationship between two participants or contain communication constructs, abstract activities, or nested abstract containers. The optional plugin type τ and the corresponding type-specific content T of an abstract connector

can define how the source and the target abstract containers C_{source} and C_{target} have to be refined, i.e., the MEPs, the derived roles for the participants, and the implementation in terms of communication protocols and middleware. During refinement of an abstract connector, communication activities are placed in the abstract containers. This is also realized by a corresponding implementation of the function δ .

3.2 Refinement Strategies

The main focus of our approach is to provide modeling constructs that span from choreography modeling to execution of the enacting workflows and to identify refinement strategies, which will be positioned within the CDC life cycle. While identifying refinement strategies and suggesting the use of some refinement mechanisms such as the refinement with process fragments, it is not our goal here to provide decisions support for finding the most suitable process fragments for refining an abstract activity or an abstract connector. That means we do not discuss and provide implementations for the function δ in this paper but leave it open for future work.

Fig. 2 shows the phases of our approach and the models and artifacts needed as input to and produced as output by each phase. Additionally, we also show the use of abstract constructs and their potential refinements throughout all phases. For every step in our approach the following is true: Abstract constructs are annotated by its modeler with information in which steps of our approach the refinement is allowed to be carried out (thus addressing requirement R6). The information defines a range of approach steps, i.e., in which step the refinement can be conducted first and in which step the refinement has to be conducted at the latest. This relates to the attributes α and ω in Definition 1. With the concept of the plugin type τ and the type-specific content T , i.e., the plugin concept, of an abstract construct, we fulfill requirement R9 since the abstract constructs are generic but provide possibilities to consider the idiosyncrasies of different domains.

The first three phases of Fig. 2 belong to the CDC Modeling phase. A domain problem (Fig. 2, (1)) is transformed into a choreography (Fig. 2, (2)) through collaborative *manual modeling* using a choreography editor. We identify the following cases where abstract constructs can be used: (i) Choreography modelers add abstract activities to define placeholders for orchestration logic that has to be concretized later during workflow refinement, deployment or execution (R1). The modelers must specify the plugin type τ and, thus, type-specific content T , e.g., preconditions and effects relating to a choreography context model (R9). (ii) Choreography modelers specify abstract activities to defer the decision if and which participant should be part of the overall choreography to a following life cycle phase (R8). (iii) A choreography modeler specifies two or more communicating participants without defining the Message Exchange Pattern [17]. That means, it is only specified that the choreography participants will exchange messages during runtime but all details are left open (R7) (e.g., if an immediate response has to follow after a request or if fault messages could be thrown). To

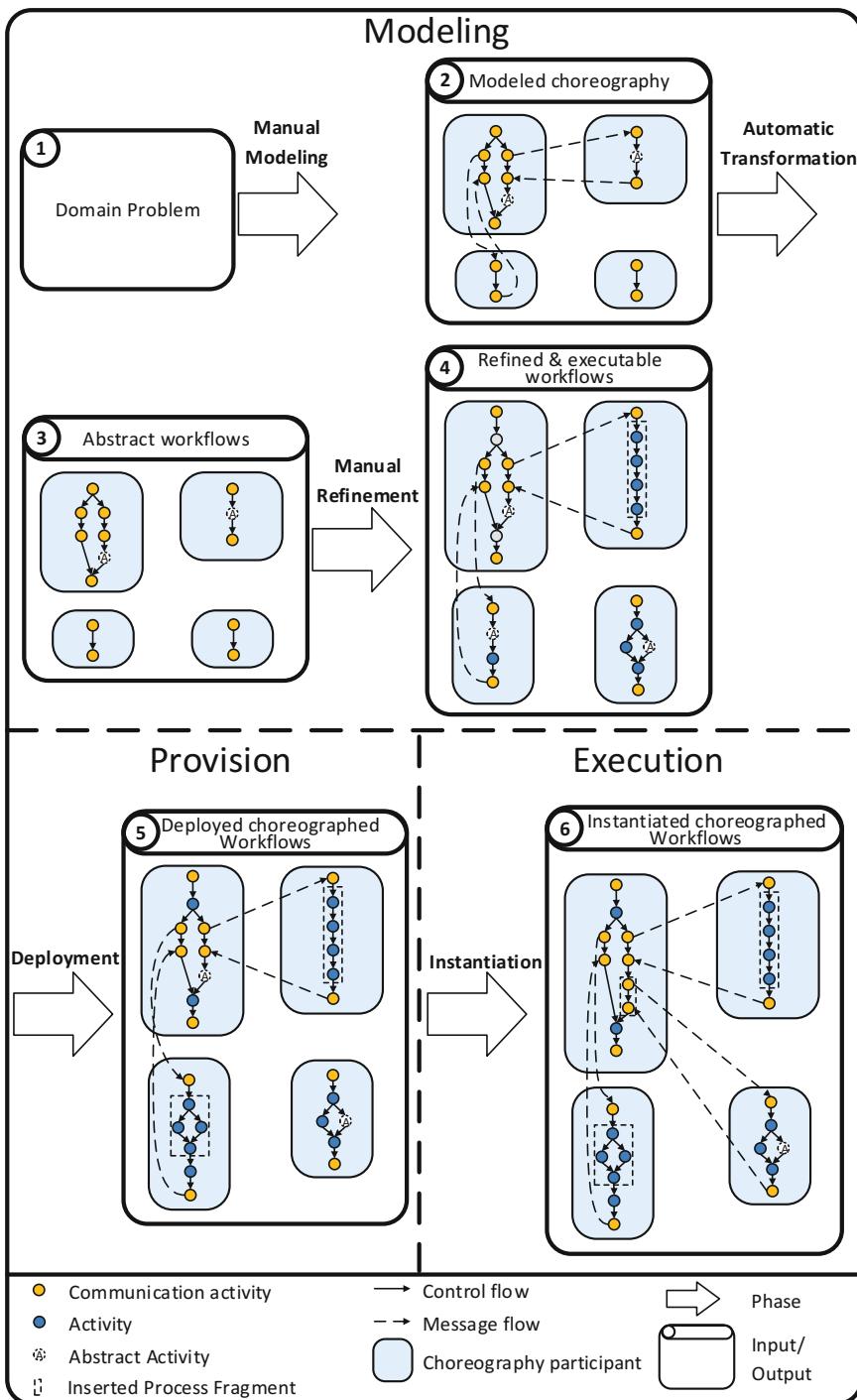


Fig. 2. Phases of the approach for enactment of flexible choreographies

realize this we use the *abstract connectors*. The plugin type of the abstract connector can be used to specify a plugin realizing a specific MEP. If the plugin type is left empty by the choreography modeler, we envision an wizard that let the user answer predefined questions to determine an appropriate refinement. Since choreography modeling languages are often not executable [8], the modeled choreography is *automatically transformed* into an abstract representation (Fig. 2, (3)) of an executable workflow language. This representation is not yet executable. For each participant a separate workflow model is generated capturing the individual communication behavior through communication constructs such as sending and receiving activities. The abstract constructs modeled in the previous phase are embedded into the abstract workflow models. The communication links represented by the abstract connectors are expressed in the existing language capabilities of the target workflow language, e.g., *Partner Links* in the case of BPEL.

In the next phase the abstract workflow models are *refined manually* by domain experts, who add orchestration logic such as backend service invocations and variable assignments thus making the models executable (Fig. 2,(4)). We identify the following refinement cases that can take place during the workflow refinement phase: (i) The abstract activities inserted in the choreography modeling phase can be refined using workflow fragments from a repository and by placing the fragments on the abstract activity in a workflow modeling editor (R2). In this case the process modeler may optionally consider the type-specific content T for the fragment selection. Alternatively, an appropriate process fragment may be selected automatically by involving a recommendation service that contacts a external context-aware decision component for this purpose. Note that the inserted process fragments may also contain abstract activities which have to be refined immediately or during workflow run time. (ii) Abstract activities of a workflow model with communication related process fragments may be inserted, i.e., fragments containing sending and receiving activities. The actual collaboration partner has to be discovered during deployment or run time. (iii) Existing abstract activities may remain unchanged. (iv) Abstract connectors can be refined by inserting the necessary communication related process fragments in the corresponding abstract containers of the collaborating workflows.

The *deployment phase* of the enacting workflows is related to the CDC Provision phase. The executable workflow models are deployed (Fig. 2, (5)) on one or more workflow engines. The refinement possibilities during deployment are: (i) Modeled abstract activities can be refined automatically through the assistance of the external context-aware decision component that is contacted by a deployment manager component in order to retrieve an appropriate process fragment. (ii) Modeled abstract activities are refined manually by asking a user to provide appropriate process fragments. (iii) Abstract connectors can be refined by automatically inserting appropriate process fragments in the involved workflows. (iv) Abstract connectors can be refined by asking a user to insert appropriate process fragments in the involved workflows.

The last phase of our approach is the *Execution* phase of the CDC system. Possible refinements in this phase are: (i) Substitution of abstract activities with fragments or executable activities automatically. Hence, partially refined workflows can be executed, too (R5). Note that the process fragment may also contain further abstract activities. (ii) Manually specification of process fragments replacing an abstract activity using a workflow editor or retrieving the fragments from a repository. This implies pausing the execution and explicitly asking a user to provide the necessary refinement by considering the current context. (iii) Refinement of abstract connectors by inserting process fragments and/or executable activities in the affected workflow instances on both sides of the connector, i.e., refining the corresponding abstract containers. Note that abstract activities can also be placed in fault handling or compensation constructs on choreography and workflow level. The fault handling or compensation activities are refined using the same strategies as described above. To ensure that a choreography is still enactable and deadlock free the resulting choreography has to be verified [15] and the correctness of the enacting workflows guaranteed. Verification mechanisms, however, are not in the scope of this paper.

4 Realization

4.1 Required Language and Tool Extensions

In order to enable the modeling of different domain problems with our approach, a choreography language and a choreography editor are needed. As a basis for a CDC system modeling tool we use our ChorDesigner [26] as choreography editor and we have chosen BPEL4Chor [9] as the choreography language to serialize the choreographies in. BPEL4Chor forms a layer on top of BPEL and specifies the participants of a choreography, their communication behavior and the message links between them. The technical details of the communication are separated from the choreography. Because BPEL4Chor itself is not executable, we transform the choreography models to abstract BPEL processes [20] and refine them with orchestration logic. For the purpose of enabling our approach, extensions of both the choreography language and the modeling tool were required. BPEL4Chor and BPEL have been extended with the concept of abstract activities. Additionally, BPEL4Chor has been extended with the concept of abstract connectors.

Extending BPEL4Chor for flexibility: The communication behavior of a choreography participant is specified in BPEL4Chor by the so-called Participant Behavior Descriptions (PBD). PBDs are based on abstract BPEL processes with a more restricting profile excluding BPEL Partner Links. To enable more flexibility when modeling choreographies, we extended the BPEL4Chor language with abstract activities using the extension mechanism of BPEL to define an extension activity denoted as *abstractActivity* in the PBD. Listing 1 shows an example of an abstract activity already taking into account the APF domain (*pluginType = APF_AdaptationManager*) embedded into a BPEL4Chor PBD. The abstract activity represents the behavior of a payment participant, which is

a part of a trip booking choreography from our motivating example in Sec. 2.2. The participant has to receive a payment request, process it and reply accordingly. Instead of specifying the payment orchestration logic, the PBD contains the extension activity `<abstractActivity>`. The *refinementStartPhase* and *refinementEndPhase* attributes indicate in this example that it can be done in the workflow modeling phase, or during deployment or run time. The refinement manner, i.e., manually or automatically, depends entirely on the plugin.

Listing 1. Extended BPEL4Chor Participant Behavior Description

```

<process name="payment" targetNamespace="urn:allowEnsembles:payment"
  abstractProcessProfile="urn:HPI_IAAS:choreography:profile:2006/12"
  xmlns:aa="urn:abstractActivity" xmlns:apf="urn:flows">
  <sequence>
    <receive wsu:id="receivePaymentRequest"/>
    <extensionActivity>
      <aa:abstractActivity name="ConductPayment"
        refinementStartPhase="workflow_modeling"
        refinementEndPhase="execution" pluginType="
          APF_AdaptationManager" failureStrategy="none">
        <aa:typeSpecificContent>
          <apf:entities/>
          <apf:preCondition/>
          <apf:effect/>
          <apf:goal/>
          <apf:compensationGoal/>
        </aa:typeSpecificContent>
      </aa:abstractActivity>
    </extensionActivity>
    <reply wsu:id="replyPaymentRequest"/>
  </sequence>
</process>
  
```

In the example of Listing 1, the plugin is of the type *APF_AdaptationManager* integrating entity, precondition, effects, goal, and compensations goal elements without specifying their details for the sake of brevity. *Abstract connectors* are realized by an extension activity in the PBDs representing source and target abstract containers of an abstract connector as well as by an extension of the BPEL4Chor Message Link concept with the attributes defined in Definition 3.

Extending BPEL for flexibility: We use the standard extensibility mechanism of BPEL and define an extension activity representing an abstract activity as a part of an executable BPEL process. By using the standard extensibility mechanism we remain compliant to the BPEL standard. The extension activity in a BPEL process has an identical structure as the one depicted in Listing 1 for the BPEL4Chor Participant Behavior Description. Furthermore, an additional extension activity represents the source and target abstract containers of an abstract connector in the BPEL workflows.

Extensions in the transformation from BPEL4Chor to BPEL: We have extended the transformation from BPEL4Chor to BPEL with the necessary mapping of the BPEL4Chor abstract activity to the BPEL abstract activity. BPEL4Chor Message Links that represent abstract connectors are transformed into BPEL Partner Links. The corresponding source and target abstract containers are transformed to their BPEL extension activity equivalent.

Plugin Concept: The realized abstract constructs specify the attributes defined formally in Sec. 3.1 and exhibit a plugin-point for type-specific content. In our Eclipse-based³ implementation of the choreography [26] and workflow editor [23], the plugins consists of an EMF data model to describe the type-specific content and a user interface extension for each plugin. The user interface extension provides input fields for populating the type-specific content of the plugin. Depending on the type-specific content, the refinement of the abstract constructs, i.e., the application of the function δ , is either done manually or automatically. In our workflow engine SCE^{MT} [12], plugins can be registered implementing automatic refinement functions for different plugin types.

4.2 Case Study

For the evaluation of our approach we use the motivating example from Sec. 2.2 to demonstrate how the modeling and manual refinement of the abstract

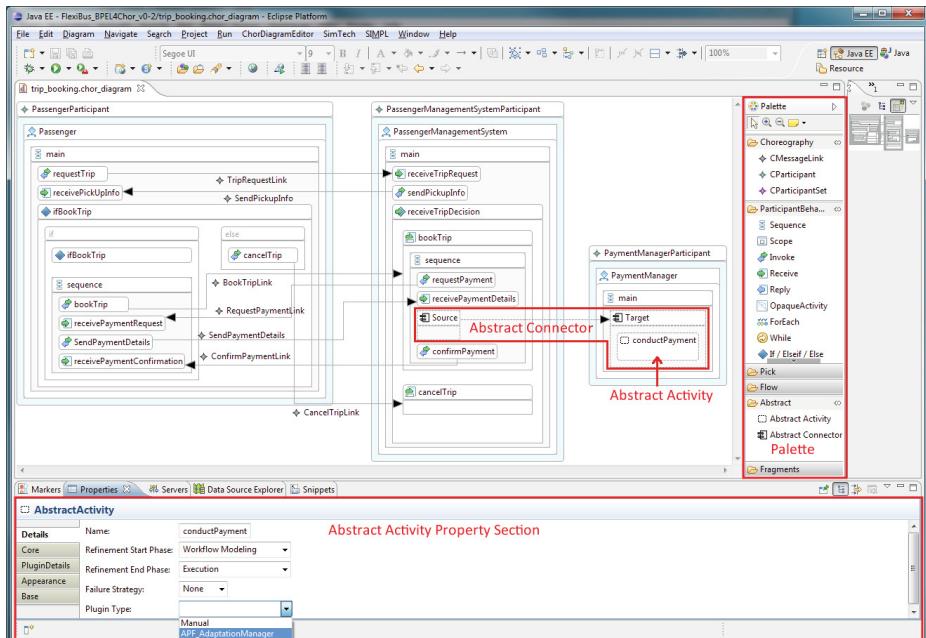


Fig. 3. Trip booking scenario modeled as choreography with our ChorDesigner

³ <http://www.eclipse.org>

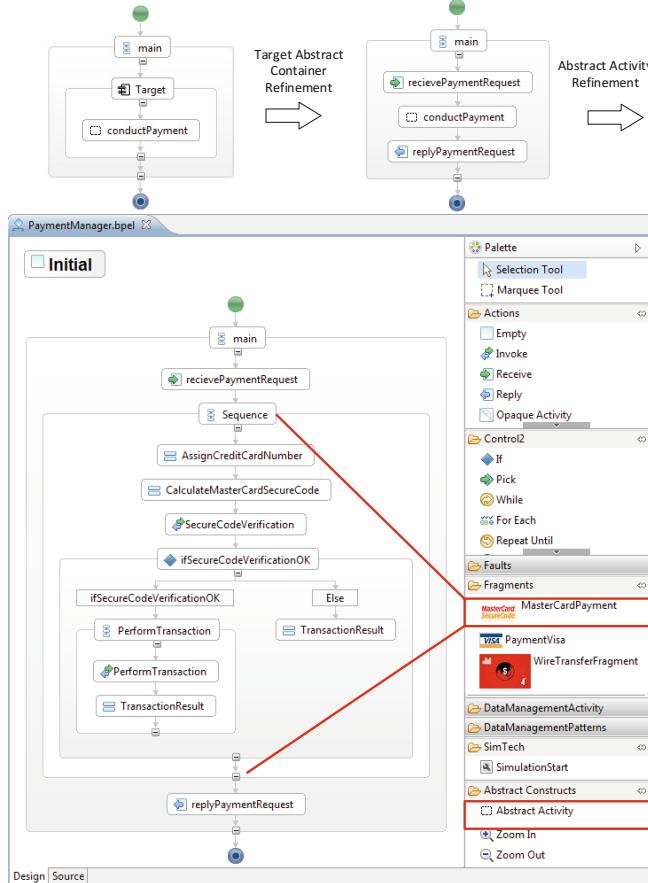


Fig. 4. Exemplary refinement of abstract constructs using the Mayflower BPEL Designer

activities and connectors have been enabled. Fig. 3 shows the trip booking scenario modeled as a choreography with the ChorDesigner. There are three participants connected with message links. Note that an abstract activity has been modeled in the *PaymentManager* participant to provide a placeholder for the actual payment logic that can be refined during later phases. Furthermore, the interaction between the *Passenger Management System* and the *Payment Manager* is not explicitly specified via message links indicating an MEP, but rather with an abstract connector allowing to defer this decision to a later point in time. The tool set presented here allows for transforming the BPEL4Chor choreography into abstract BPEL processes [26]. All newly introduced choreography level abstract constructs are transformed into the corresponding constructs on the orchestration level; our system also generates valid WSDL interfaces. The refinement of abstract BPEL processes is conducted with the Mayflower BPEL

Designer [23] and Fig. 4 shows some possible refinement steps of the generated BPEL process of the PaymentManager participant. First, the target abstract container is refined with communication activities, then the *conductPayment* abstract activity is replaced by a credit card process fragment. Additionally, our workflow management system is capable of refining abstract activities during run time on the workflow level [5] as well as arbitrary manual insertion of process fragments during run time [24]. Not yet realized but planned for the near future is the run time refinement of abstract connectors and abstract activities that implies that new participants are joining the choreography.

5 Conclusions and Future Work

In this work we have introduced an approach that uses abstract constructs (abstract activities and abstract connectors) and refinement strategies as means to provide end-to-end flexibility to Collaborative, Dynamic & Complex systems and their realization in terms of choreographies and enacting workflows. The approach permits the use of abstract constructs and their refinement to concrete logic – manually or automatically – throughout the whole life cycle of CDC systems. With this approach modeling decisions can be deferred to later points in the CDC life cycle – a requirement from different domains where CDC systems are applied, which has not been addressed by literature yet. With this approach we also support the use of many existing approaches for modeling and/or generation of concrete logic on the level of the enacting orchestrations. We plan to investigate the run time refinement of abstract activities and connectors on the level of choreographies and address the case of introducing new choreography participants during execution. Run time changes of choreographies and their propagation to the enacting workflows are also part of our future research.

Acknowledgment. This work is funded by the projects FP7 EU-FET 600792 ALLOW Ensembles and the German DFG within the Cluster of Excellence (EXC310) Simulation Technology.

References

1. 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)
2. Andrikopoulos, V., Bucchiarone, A., Gómez Sáez, S., Karastoyanova, D., Mezzina, C.A.: Towards Modeling and Execution of Collective Adaptive Systems. In: Lomuscio, A.R., Nepal, S., Patrizi, F., Benatallah, B., Brandić, I. (eds.) ICSOC 2013. LNCS, vol. 8377, pp. 69–81. Springer, Heidelberg (2014)
3. Andrikopoulos, V., Gómez Sáez, S., Karastoyanova, D., Weiβ, A.: Collaborative, Dynamic & Complex Systems: Modeling, Provision & Execution. In: CLOSER 2014, p. 10. SciTePress (2014)

4. Ardissono, L., Furnari, R., Goy, A., Petrone, G., Segnan, M.: A framework for the management of context-aware workflow systems. In: WEBIST (1), pp. 80–87 (2007)
5. Bialy, L.: Dynamic Process Fragment Injection in a Service Orchestration Engine. Diploma Thesis No. 3564, University of Stuttgart, Germany (2014)
6. Buccharone, A., Marconi, A., Pistore, M., Raik, H.: Dynamic Adaptation of Fragment-Based and Context-Aware Business Processes. In: ICWS 2012, pp. 33–41. IEEE (2012)
7. Buccharone, A., Lafuente, A.L., Marconi, A., Pistore, M.: A formalisation of adaptable pervasive flows. In: Laneve, C., Su, J. (eds.) WS-FM 2009. LNCS, vol. 6194, pp. 61–75. Springer, Heidelberg (2010)
8. Decker, G., Kopp, O., Barros, A.: An Introduction to Service Choreographies. Information Technology 50(2), 122–127 (2008)
9. Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4Chor: Extending BPEL for Modeling Choreographies. In: ICWS 2007. IEEE (2007)
10. Fdhila, W., Rinderle-Ma, S., Reichert, M.: Change Propagation in Collaborative Processes Scenarios. In: CollaborateCom 2012. IEEE (2012)
11. Guo, C.J., Sun, W., Huang, Y., Wang, Z.H., Gao, B.: A Framework for Native Multi-Tenancy Application Development and Management. In: CEC/EEE 2007, pp. 551–558. IEEE (2007)
12. Hahn, M.: Approach and Realization of a Multi-tenant Service Composition Engine. Diploma Thesis No. 3546, University of Stuttgart, Germany (2013)
13. Herrmann, K., Rothermel, K., Kortuem, G., Dulay, N.: Adaptable Pervasive Flows - An Emerging Technology for Pervasive Adaptation. In: SASOW 2008, pp. 108–113. IEEE (2008)
14. Krebs, R., Momm, C., Kounev, S.: Architectural Concerns in Multi-tenant SaaS Applications. In: CLOSER 2012, pp. 426–431. SciTePress (2012)
15. Lohmann, N., Kopp, O., Leymann, F., Reisig, W.: Analyzing BPEL4Chor: Verification and participant synthesis. In: Dumas, M., Heckel, R. (eds.) WS-FM 2007. LNCS, vol. 4937, pp. 46–60. Springer, Heidelberg (2008)
16. Mahfouz, A., Barroca, L., Laney, R., Nuseibeh, B.: Requirements-Driven Collaborative Choreography Customization. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 144–158. Springer, Heidelberg (2009)
17. Nitzsche, J., van Lessen, T., Leymann, F.: Extending BPEL light for Expressing Multi-Partner Message Exchange Patterns. In: EDOC 2008, pp. 245–254. IEEE (2008)
18. OASIS: Web services business process execution language version 2.0 (April 2007), <http://docs.oasis-open.org/ws-bpel/2.0/ws-bpel-v2.0.html>
19. Reichert, M., Rinderle-Ma, S., Dadam, P.: Flexibility in process-aware information systems. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 115–135. Springer, Heidelberg (2009)
20. Reimann, P.: Generating BPEL Processes from a BPEL4Chor Description. Student Thesis No. 2100 (2007)
21. Rinderle, S., Wombacher, A., Reichert, M.: Evolution of Process Choreographies in DYCHOR. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 273–290. Springer, Heidelberg (2006)
22. Sadiq, S., Sadiq, W., Orlowska, M.: Pockets of Flexibility in Workflow Specification. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, pp. 513–526. Springer, Heidelberg (2001)

23. Sonntag, M., Hahn, M., Karastoyanova, D.: Mayflower - Explorative Modeling of Scientific Workflows with BPEL. In: CEUR Workshop 2012, pp. 1–5. Springer (2012)
24. Sonntag, M., Karastoyanova, D.: Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows. Grid Computing 11(3), 553–583 (2013)
25. 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)
26. Weiß, A., Andrikopoulos, V., Gómez Sáez, S., Karastoyanova, D., Vukojevic-Haupt, K.: Modeling Choreographies using the BPEL4Chor Designer: an Evaluation Based on Case Studies. Technical Report 2013/03, University of Stuttgart (2013)
27. Wombacher, A.: Alignment of Choreography Changes in BPEL Processes. In: SCC 2009, pp. 1–8. IEEE (2009)

Business Process Fragments Behavioral Merge

Mohamed Anis Zemni¹, Nejib Ben Hadj-Alouane², and Amel Mammar³

¹ Université de Tunis El Manar, Ecole Nationale des Sciences de l'Informatique,
Ecole Nationale d'Ingnieurs de Tunis, UR/OASIS, 1002 Tunis, Tunisia
mohamedaniszemni@gmail.com

² Université de Tunis El Manar, Ecole Nationale d'Ingnieurs de Tunis, ENIT,
UR/OASIS , 1002 Tunis, Tunisia
nejib_bha@yahoo.com

³ Institut Mines-Télécom/Télécom SudParis, CNRS UMR 5157 SAMOVAR, France
amel.mammar@telecom-sudparis.eu

Abstract. In the present work, we propose an approach to merge business process fragments in order to facilitate the reuse of fragments in business process designs. The approach relies on the so-called adjacency matrices. Typically used to handle graphs, this concept represents a new way to systematically merge fragments through their corresponding matrices. At the same time, fragments merging must keep the behavior of the original fragments consisting of their execution scenarios and rule out undesirable ones that may be generated during the merge task. Indeed, such behaviors probably lead to process execution blocking. The proposed approach has been implemented and tested on a collection of fragments and experimental results are provided.

Keywords: Business process fragment, merge, behavior, adjacency matrix.

1 Introduction

With the emergence of new Web technologies, companies are keen to organize their businesses in a flexible way, in order to face the tough competition. Optimizing the development periods of service-oriented software is a key strategy that can positively influence the productivity and efficiency of the various actors in software industry. Services accessible through the various software artifacts, available nowadays, are mostly associated with workflows or so-called *business processes* [1]. Providing good designs for these processes may, therefore, improve the quality of the services they eventually deliver.

A business process development approach solely based on constructing business processes completely from scratch can be easily qualified as lacking flexibility. In fact, for maximum effectiveness, designers need to implement, optimize and test new software in short periods of time. To help overcome such hurdles, business process designers and architects are now interested in *past user experiences*, as demonstrated, for example, in the Web service discovery [2]. The reuse and integration of existing processes into the new ones, rather than completely

building them from scratch, can be beneficial and can increase both efficiency and productivity [3]. A new and more practical approach, that is emerging, consists in reusing some selected *fragments*, i.e., sub-processes, from existing processes; these fragments are constrained with fewer business rules than their complete parent processes and are, hence, easier to integrate into new processes *in order to fulfill a given specified functionality*. The controlled reuse of well-established (tested and stable) artifacts, as demonstrated in the literature, can improve the quality of newly developed processes [4,5].

However, during the building of fragment-based processes, some selected fragments may be delivered with similar or even overlapping knowledge and structures. Overlapping fragments consist of a set of similar process elements occurring inside these fragments. This may be explained by the fact that, initially, fragments were independently retrieved from several processes. The commonalities between fragments are typically used to connect these fragments. Moreover, fragments must be well structured so as to enable executing them inside complete business processes. Therefore, process designers are keen to find the best way to assemble overlapping fragments with each other while building new processes. At the same time, the merged fragments must ensure that the behavior of the fragments, i.e., that are obtained from the merge, subsumes the original fragment behavior. Indeed, after performing the merge task, new paths may be added, thus extending the behavior into undesirable ones. Consequently, the obtained fragments must be provided with behavioral constraints.

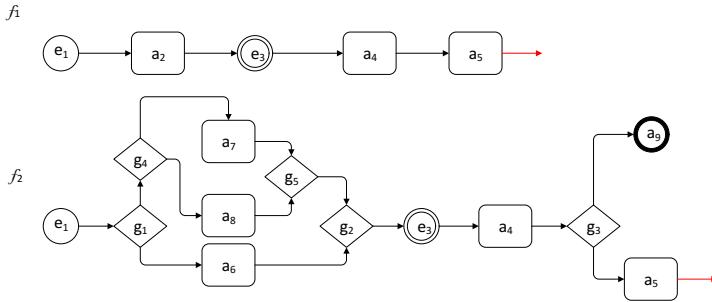
The paper is articulated as follows. In section 2, we present the business process fragment model and its corresponding paths; main concepts that are needed to present our approach. We also state the merge issues that are related to preserving the behavior of the original fragments. We then introduce the concept, that our merge mechanism revolves around, in section 3. More specifically, we present the manner of adapting adjacency matrices, typically used in graph domain, in order to handle fragments. We also provide a set of properties that each fragment matrix must follow. In section 4, we describe our matrix-based merge mechanism as well as some rules that permit to systematically derive the corresponding merged fragment matrix. We also propose to annotate the resulting fragments with constraints to keep the original fragment behavior. Experimental results are presented in section 5 and related works are presented in section 6. We finally conclude our paper in section 7.

2 Business Process Fragment Behaviors

In the following, we present the inputs of our approach, namely business process fragments as well as the behavioral issues that the merge task can be subject to.

2.1 Business Process Fragments and Paths

A business process fragment (fragment for short) is a sub-process describing an incomplete knowledge. It is defined in the literature ([6,7]) as a *connected* portion

**Fig. 1.** Input Fragments Represented in BPMN

of a business process that is designed for *reuse* purposes. Generally speaking, a fragment has relaxed completeness criteria [8] compared to complete business processes. We recall that a business process is complete when all required activities are fully specified. A fragment is composed of at least one object, and of several edges, representing control flows. Objects are essentially made of activities, events, and gateways. Inspired from the BPMN representation [9], the latter constructs are respectively represented with rounded boxes, circles, and diamonds (as shown in Figure 1). Simple circles depict start events, double circles illustrate intermediate events, and thick circles represent end events. Control flows represent the *dependency* among the objects, namely their execution order. They are represented with solid arrows. Activities involve tasks to perform, events involve waiting tasks, and gateways are routing constructs used to control the divergence of control flows, i.e., when a control flow has to be split into a set of control flows, or their convergence, i.e., when a set of control flows have to be joined into a single one [10]. When a control flow involves a gateway as a source object, a passing condition must be provided to indicate which branch should be followed. A fragment is intended to be composed with other ones to build new complete processes. In contrast to complete business processes, fragments can depict dangling control flows, i.e., with either no source or target activities (source and target for short) specified. Dangling control flows (depicted in Figure 1 in red) represent gluing points from which they would be attached to other fragments in order to build complete business processes [11]. A fragment model, from our point of view, is formally defined as follows:

Definition 1 (Business Process Fragment). A Business Process Fragment is a tuple $f = (O, A, G, \text{type}, C_f, \chi)$, where

- $O \subseteq (A \cup G)$ is a set of objects composed of a set of activities, A , and a set of gateways, G ,
- $\text{type} \in G \rightarrow \{\text{'div'}, \text{'conv'}\}$ returns the type of a given gateway,
- $C_f \subseteq ((O \cup \{\perp\}) \times (O \cup \{\perp\})) - \{(\perp, \perp)\}$ is the complete and/or dangling control flow relation to link objects to each other, where symbol \perp represents the missing source or target of a control flow. C_f is such that:

- each activity or event, a , can have at most one incoming (resp. outgoing) control flow. Formally, $\forall a. (a \in A \Rightarrow |C_f[\{a\}]| \leq 1 \wedge |C_f^{-1}[\{a\}]| \leq 1)$,
- each gateway is either a divergence or a convergence. A divergence (resp. convergence) gateway has one incoming (resp. outgoing) control flow and several outgoing (resp. incoming) control flows: $\forall g. (g \in G \Rightarrow (\text{type}(g) = \text{'div'} \wedge |C_f[\{g\}]| \geq 1 \wedge |C_f^{-1}[\{g\}]| = 1) \vee (\text{type}(g) = \text{'conv'} \wedge |C_f[\{g\}]| = 1 \wedge |C_f^{-1}[\{g\}]| \geq 1))$
- $\chi : G \triangleleft C_f^{-1} \rightarrow \text{cond}$ is a function that maps control flows involving a gateway as a source to a passing condition, cond^2 .

Note that activities and events are set in the same group. In the rest of the paper, we use “activities” instead of “activities and events”. When a gateway is involved as a source object in a control flow whose passing condition is not yet defined, then the gateway is denoted γ and called configurable gateway. Moreover, such passing conditions are written with the symbol ‘?’ . Note that configurable gateways as well as related passing conditions are configured at build time by the process designer.

This paper is about merging a set of fragments’ behaviors related to the execution of a set of paths that we call control flow paths. In a business process, a *control flow path* (path for short), is an alternated control flow sequence between two *objects*, where the target object of one control flow is the source object of the following control flow. A *gateway path* describes a path connecting a couple of *activities*, i.e. a source and a target activities, by means of a control flow, or traversing a sequence of gateways. They are formally defined as follows.

Definition 2 (Control Flow Paths and Gateway Paths). Given a fragment $f = (O, A, G, \text{type}, C_f, \chi)$, we define a path between a couple of objects $a \in O$ and $b \in O$, respectively referred to as source and target objects, as a sequence of alternated control flows $\langle (o_i, o_{i+1}) \in C_f \rangle_{i \in 1..n-1}$, where $o_1 = a$ and $o_n = b$ and $o_i \in O$. A path is called a gateway path iff, $o_1 \in A$ and $o_n \in A$, and $o_i \in G$, for each i such that $2 \leq i \leq n - 1$.

Note that in our work, we call a couple of activities *consecutive*, if they are linked with a gateway path. For a given fragment f , let Π be the set of control flow paths and Π_G be the set of gateway paths. In addition to the above definition, we define the function $P_i(p)$ which returns the i^{th} control flow in a path, p , where $1 \leq i \leq p.\text{length}^3$ (e.g., $P_1(\langle (e_1, a_2), (a_2, e_3) \rangle) = (e_1, a_2)$).

Note that several paths may link two distinct objects while we assume that there can only be a single gateway path between two consecutive activities. We define the function $P : O \times O \rightarrow \mathcal{P}(\Pi)^4$ that returns the set of paths linking a couple of objects, and the function $P_G : A \times A \rightarrow \Pi_G \cup \{\text{null}\}$ that returns the

¹ Given $R \subseteq X \times Y$, then $X_1 \triangleleft R = \{(x, y) | (x, y) \in R \wedge x \in X_1\}$.

² The outgoing control flow gets *true* if the corresponding condition is evaluated to *true*.

³ $S.\text{length}$ depicts the number of elements of the sequence S .

⁴ $\mathcal{P}(S)$ depicts the power set of a set S .

gateway path linking a couple of consecutive activities, where *null* means that the considered activities are not consecutive. For instance, in the fragment f_2 of Figure 1, $P(e_1, g_5) = \{(e_1, g_1), (g_1, g_4), (g_4, a_7), (a_7, g_5)\}$, $\{(e_1, g_1), (g_1, g_4), (g_4, a_8), (a_8, g_5)\}\}$ is a set of paths between the source object e_1 and the target object g_5 . $P_G(e_1, a_6) = \langle (e_1, g_1), (g_1, a_6) \rangle$ depicts a gateway path between the source activity e_1 and the target activity a_6 . $P_G(e_1, e_3) = \text{null}$ as activities e_1 and e_3 are not consecutive.

In the following, we present the result of a couple of fragments merge and explain the reasons why we need to fix some behavioral constraints.

2.2 Business Process Fragment Behavior

When merging a couple of fragments, the resulting fragment must ensure that the behaviors of the input fragments are preserved. However, some undesirable behaviors can probably be generated during the merge task. Let us consider the merge basis proposed by Assy et al. [12] and La Rosa et al. [13]. A unique version of common regions; a set of common and connected gateway paths, is first created into the merge fragment. A couple of gateway paths between a source and a target activities separated by different gateways are considered as common. Indeed, existing gateways turn into configurable ones when they are merged. After that individual paths, either incoming to or outgoing from the common region, are inserted: paths that are incoming to (resp. outgoing from) an activity of a common region are connected to it using a convergence (resp. diverging) configurable gateway. For instance, Figure 2 depicts the resulting fragment, f_{12} , from the merge of the fragments f_1 and f_2 that are depicted in Figure 1. Note that the gateway γ_{d1} is configurable and substitutes the gateway g_3 in fragment f_2 .

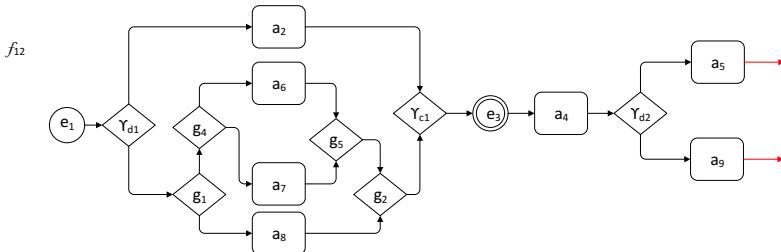


Fig. 2. Merge Result of Fragments Depicted in Figure 1

The paths of each input fragment are transferred into the merged fragment and new divergence and/or convergence gateways are added when two paths retrieved from several fragments share the same activities. For instance, the path $\langle (a_4, e_5) \rangle$ retrieved from the fragment f_1 and the path $\langle (a_4, g_3), (g_3, a_9) \rangle$ retrieved from fragment f_2 both share the activity a_4 . During the merge, a divergence configurable gateway, γ_{d2} , is added between the activities a_4 and e_5 ,

and a_4 and g_3 . This means that, in the resulting fragment, there are two options regarding the path to take during the execution. This depends on the passing condition that the designer would fix for the control flow involving the divergence gateway γ_{d2} as a source object. Given that defining passing conditions is left to the designer when they are configurable, this task is error prone since the designer may make mistakes in defining control flow passing conditions which may lead to behavior inconsistencies. For example, assume the following scenario. The merged fragment is integrated as part of a complete business process. During the execution, the passing condition of the control flow involving the configurable gateway γ_{d1} as a source object is evaluated to *true* which leads the execution to the activity a_2 . The activity a_2 is executed and so on until reaching the configurable gateway γ_{d2} . The passing condition of the control flow involving the configurable gateway γ_{d2} as a source object is evaluated to *true* leading the execution to the activity a_9 . This scenario may lead to the failure of the execution of activity a_9 when it strongly depends on the execution of activities a_6 , a_7 , or a_8 that were not eventually executed. For instance, the activity a_9 may perform "*Blood Analysis*" while a_6 , a_7 , and a_8 cooperate, individually or jointly to perform "*Blood Extracting*". This problem was never asked in the fragment f_1 as the activity a_9 can be performed only if the previous activities are executed, namely a_6 , a_7 , or a_8 . Therefore, when merging a couple of fragments, we have to ensure that each activity is executed only if at least the precedent activities in the original fragment are executed.

While the work by Assy et al. [12] does not prevent such issues, La Rosa et al. [13] annotates all control flows with their original input fragments. In this work, we show that not all control flows need to be annotated. By inspection, we can see that the original behaviors have changed after inserting configurable gateways. Indeed, in a merged fragment, we distinguish three types of paths: (i) paths that are *shared* by multiple fragments (ii) *individual* paths that come before a convergence configurable gateway, and (iii) *individual* paths that come after a divergence configurable gateway. Depicted in figure 2, the path $\langle(e_3, a_4)\rangle$ is common to both fragments f_1 and f_2 , the paths $\langle(a_2, \gamma_{c1})\rangle$ and $\langle(g_2, \gamma_{c1})\rangle$ are individual ones, respectively originating from fragments f_1 and f_2 , that come before the convergence configurable gateway γ_{c1} , and the paths $\langle(\gamma_{d2}, a_5)\rangle$ and $\langle(\gamma_{d2}, a_9)\rangle$ are individual ones, respectively originating from fragments f_1 and f_2 , that come after the divergence gateway γ_{d2} . Control flows of common paths do not need to be annotated as they will be, surely, taken during the execution, regardless of the execution of the precedent activities. Control flows of individual paths that come before a convergence configurable gateway are neither relevant to annotate as navigating between such control flows supposes that we have already chosen the right path. However, when moving to an individual path from a common path through a divergence configurable gateway, we have to make sure that the paths that are represented before that individual path in the original fragment are already taken. More specifically, a path, originating from an input fragment, that is stated after a divergence configurable gateway, in the merged fragment, is conditioned by following the paths originally retrieved from the

same input fragment that figure before a convergence configurable gateway in the merged fragment. For instance, the path $\langle (a_4, \gamma_{d1}), (\gamma_{d1}, a_9) \rangle$ depicted in Figure 2 is conditioned by the incoming paths to g_2 . Consequently, control flows belonging to paths of an input fragment and involving a *divergence configurable gateway as a source object* is conditioned by the *last control flow* involving a *convergence configurable gateway as a target object* of a path of the same fragment. Note that behavioral constraints may be represented within model note boxes [13] or as passing conditions of control flows involving gateways as source objects (as shown in section 4.3).

In the following, we present a systematic approach using matrices to merge a set of input fragments while keeping their original behavior.

3 Fragment Adjacency Matrix

In graph theory [14], a graph can be modeled with an adjacency matrix to represent the edges between two adjacent nodes. A graph adjacency matrix is a d square matrix, where d is the number of graph nodes. In our work, a fragment is a directed graph where nodes are activities. A couple of activities (respectively source and target) are considered as adjacent if they are linked by means of a gateway path. Therefore, each fragment can be mapped onto an adjacency matrix that we call fragment adjacency matrix, formally defined as follows.

Definition 3 (Fragment Adjacency Matrix). *Given a fragment $f = (O, A, G, type, C_f, \chi)$ and its corresponding gateway path set Π_G , $M_f = (A, \mu)$ is its corresponding adjacency matrix, such that A is the set of activities the matrix is defined over, and $\mu : A \times A \rightarrow \Pi_G \cup \{null\}$ is a function that returns the element gateway path which links the elements involved source and target activities. It returns null otherwise.*

	e_1	a_2	e_3	a_4	a_5
e_1		$\langle (e_1, a_2) \rangle$			
a_2			$\langle (a_2, e_3) \rangle$		
e_3				$\langle (e_3, a_4) \rangle$	
a_4					$\langle (a_4, a_5) \rangle$
a_5					

Fig. 3. Fragment Adjacency Matrix M_{f_1} for Fragment f_1 in Figure 1

Note that, in our work, matrices are represented with a table and empty cells refer to *null* values. For instance, Figures 3 and 4 depict fragment adjacency matrices (fragment matrices for short) respectively for fragments f_1 and f_2 that are illustrated in Figure 1.

Given the definition 3, we derive the following well-formedness properties that every fragment matrix must ensure, i.e., these properties being derived from Definition 1 where an activity has at most an incoming control flow and an outgoing control flow:

	e_1	e_3	a_4	a_5	a_6	a_7	a_8	a_9
e_1					$\langle\langle e_1, g_1 \rangle\rangle, \langle\langle g_1, a_6 \rangle\rangle$	$\langle\langle e_1, g_1 \rangle\rangle, \langle\langle g_1, g_4 \rangle\rangle, \langle\langle g_4, a_7 \rangle\rangle$		
e_3			$\langle\langle e_3, a_4 \rangle\rangle$					
a_4					$\langle\langle a_4, g_3 \rangle\rangle, \langle\langle g_3, a_5 \rangle\rangle$			$\langle\langle a_4, g_3 \rangle\rangle, \langle\langle g_3, a_9 \rangle\rangle$
a_5								
a_6		$\langle\langle a_6, g_2 \rangle\rangle, \langle\langle g_2, e_3 \rangle\rangle$						
a_7		$\langle\langle a_7, g_5 \rangle\rangle, \langle\langle g_5, g_2 \rangle\rangle, \langle\langle g_2, e_3 \rangle\rangle$						
a_8		$\langle\langle a_8, g_5 \rangle\rangle, \langle\langle g_5, g_2 \rangle\rangle, \langle\langle g_2, e_3 \rangle\rangle$						
a_9								

Fig. 4. Fragment Adjacency Matrix M_{f_2} for Fragment f_2 in Figure 1

- When an element contains a gateway path of length 1, then the rest of the row and the column elements must be empty. This means that the corresponding source and target activities are linked by means of a single control flow.

Property 1. Given a fragment matrix $M_f = (A, \mu)$ then, $\forall(a, b). (a \in A \wedge b \in A \wedge \mu(a, b).length = 1 \Rightarrow \forall c. (c \in A - \{a, b\} \Rightarrow \mu(a, c) = null \wedge \mu(c, b) = null))$.

For instance, the fragment matrix M_{f_2} , depicted in Figure 4, contains an element $\mu_{f_2}(e_3, a_4)$ that describes a gateway path of length 1, $\langle\langle e_3, a_4 \rangle\rangle$. The rest of the elements of the row and the column are empty.

- When at least two elements of a given row are not empty, then the first control flow of each element should be shared by all of them. This means that there are several paths going out of the corresponding source activity and are connected by means of a shared divergence gateway.

Property 2. Given a fragment matrix $M_f = (A, \mu)$ then, $\forall(a, b, c). (a \in A \wedge b \in A \wedge c \in A \wedge \mu(a, b) \neq null \wedge \mu(a, c) \neq null \Rightarrow P_1(\mu(a, b)) = P_1(\mu(a, c)))$.

For instance, the elements $\mu_{f_2}(a_4, a_5)$ and $\mu_{f_2}(a_4, a_9)$ in the fragment matrix M_{f_2} of Figure 4 contain paths sharing the first control flow (a_4, g_3).

- When at least two elements of a given column are not empty, then the last control flow of each element should be shared by all of them. This means that there are several paths coming to the corresponding target activity and are connected by means of a shared convergence gateway.

Property 3. Given a fragment matrix $M_f = (A, \mu)$ then, $\forall(a, b, c). (a \in A \wedge b \in A \wedge c \in A \wedge \mu(b, a) \neq null \wedge \mu(c, a) \neq null \Rightarrow P_{\mu(b,a).length}(\mu(b, a)) = P_{\mu(c,a).length}(\mu(c, a)))$.

For instance, the elements $\mu_{f_2}(a_6, e_3)$, $\mu_{f_2}(a_7, e_3)$, $\mu_{f_2}(a_8, e_3)$ in the fragment matrix M_{f_2} of Figure 4 contain paths sharing the last control flow (g_2, e_3).

Similarly to typical adjacency matrices, a fragment matrix describes a set of *walks* leading from a given source activity, i.e., represented in a row, to a target activity, i.e., represented in a column, in an alternated fashion. A walk is formally defined as follows.

Definition 4 (Walk). *Given a fragment matrix $M_f = (A, \mu)$, a walk is an alternated succession of non empty elements $\langle \mu(a_1, a_2), \mu(a_2, a_3) \dots \mu(a_{n-2}, a_{n-1}), \mu(a_{n-1}, a_n) \rangle$ from activity a_1 leading to activity a_n , where $a_i \in A$.*

For instance, $\langle \mu_{f_1}(e_1, a_2), \mu_{f_1}(a_2, e_3), \mu_{f_1}(e_3, a_4) \rangle$ is a walk from activity e_1 leading to activity a_4 , in the matrix M_{f_1} depicted in Figure 3. Note that a walk between a couple of activities is a path.

4 Behavior-Aware Fragments' Adjacency Matrices Merge

In this section, we present the manner to merge fragments using matrices. We then propose an algorithm to retrieve the necessary annotations to keep the input fragments' behaviors. Given a couple of fragment matrices, the merge consists in merging elements' gateway paths of both matrices. We then apply some rules to transform the merged matrix into a fragment matrix.

4.1 Gateway Paths Merge

In [12], the merge of a couple of gateway paths is applied over a couple of gateway paths that share the same source and target activities. It consists in merging gateways of one gateway path with gateways of the other one in order to generate a single gateway path connecting the source activity to the target activity. In our work, we adapt the merge mechanism to control flows. When merging a couple of gateway paths, each control flow from one gateway path is merged with a control flow of the same index⁵ from the other gateway path. To enable merging a couple of gateway paths having the same length, $p_1 = \langle (o_i, o_{i+1}) \rangle_{i \in 1..n-1}$ and $p_2 = \langle (o'_i, o'_{i+1}) \rangle_{i \in 1..n-1}$, the following conditions should be satisfied:

1. both gateway paths must have the same source and target activities: $o_1 = o'_1$ and $o_n = o'_n$,
2. at the same position, both paths have the same gateway's type: $type(o_i) = type(o'_i)$, with $2 \leq i \leq n - 1$.

Given that in some cases, gateway paths do not follow the above conditions, gateway paths should first be aligned in order to enable the associated fragments merge. A gateway paths' alignment consists in inserting a set of configurable gateways, either convergence, γ_c , or divergence, γ_d , ones, inside the gateway paths in order to equalize their length and unify the gateway types. The gateway paths alignment is formally defined as follows.

⁵ Given a sequence of elements $E = \langle e_1, e_2, \dots, e_n \rangle$, e_i is an element having index i .

Definition 5 (Gateway Paths Alignment). *The gateway paths alignment of a couple of gateway paths, $p_1 = \langle(o_i, o_{i+1})\rangle_{i \in 1..n-1}$ and $p_2 = \langle(o'_j, o'_{j+1})\rangle_{j \in 1..m-1}$, where $o_1 = o'_1$ and $o_n = o'_m$, is defined as their transformation into aligned gateway paths $p_1^A = \langle(o_k, o_{k+1})\rangle_{k \in 1..q-1}$ and $p_2^A = \langle(o'_k, o'_{k+1})\rangle_{k \in 1..q-1}$ having the same length, such that $\text{type}(o_k) = \text{type}(o'_k)$, with $k \in 2..(q-1)$.*

For instance, the gateway paths $\langle(a_4, a_5)\rangle$ and $\langle(a_4, g_3), (g_3, a_5)\rangle$ respectively retrieved from the fragments f_1 and f_2 are respectively aligned into $\langle(a_4, \gamma), (\gamma, a_5)\rangle$ and $\langle(a_4, g_3), (g_3, a_5)\rangle$, where $\text{type}(\gamma) = \text{type}(g_3) = \text{div}'$.

Let us notice that finding the optimal alignment is out of the scope of this paper and readers may refer to the work in [12] to get more details. Once the gateway paths are aligned, they can be merged as formally described by the following definition.

Definition 6 (Gateway Paths Merge). *Given a couple of aligned gateway paths $p_1^A = \langle(o_k, o_{k+1})\rangle_{k \in 1..q-1}$ and $p_2^A = \langle(o'_k, o'_{k+1})\rangle_{k \in 1..q-1}$, where $o_1 = o'_1$ and $o_q = o'_q$, their merge, written $p_1^A \oplus p_2^A$, returns a single gateway path $p^A = \langle(o''_k, o''_{k+1})\rangle_{k \in 1..q-1}$, where:*

- $o''_k = o_k = o'_k$, for $k \in \{1, q\}$,
- $\text{type}(o''_k) = \text{type}(o_k) = \text{type}(o'_k)$, for $k \in 2..(q-1)$,
- $\chi((o''_k, o''_{k+1})) = \chi((o_k, o_{k+1})) \bullet \chi((o'_k, o'_{k+1}))$, for $k \in 2..(q-1)$, where \bullet is either \wedge or \vee .

4.2 Fragments' Adjacency Matrices Merge

Given Definition 6, the merge of a couple of fragment matrices consists in merging elements sharing the same source and target activities. Note that Matrices element gateway paths are first aligned, as demonstrated in Definition 5. The matrices merge is formally defined as follows:

Definition 7 (Fragment Adjacency Matrices Merge).

Given a couple of fragment matrices, M_{f_1} and M_{f_2} , corresponding to fragments f_1 and f_2 , their merge matrix is a matrix $M = (A, \mu)$, such that $A = f_1.A \cup f_2.A$, and $\forall a_i, a_j \in M.A$

$$\mu(a_i, a_j) = \begin{cases} \mu_{f_1}(a_i, a_j) & , \text{ if } \{a_i, a_j\} \subseteq M_{f_1}.A \text{ and } \{a_i, a_j\} \not\subseteq M_{f_2}.A \\ \mu_{f_2}(a_i, a_j) & , \text{ if } \{a_i, a_j\} \subseteq M_{f_2}.A \text{ and } \{a_i, a_j\} \not\subseteq M_{f_1}.A \\ \mu_{f_1}(a_i, a_j) \oplus \mu_{f_2}(a_i, a_j) & , \text{ if } \{a_i, a_j\} \subseteq M_{f_1}.A \cap M_{f_2}.A \\ \text{null} & , \text{ otherwise} \end{cases}$$

For instance, the matrix M depicted in Figure 5 represents the merge of fragment matrices M_{f_1} and M_{f_2} , respectively in Figure 3 and Figure 4.

However, the resulting matrix is not necessarily a fragment matrix. For instance, the resulting matrix, depicted in Figure 5, has broken all the properties that are stated above. Indeed, the column corresponding to the activity e_3 has

broken property 1. Indeed, the element $\mu(e_2, e_3)$ contains a path of length 1 and there exist other non empty elements in the same column. The same column also breaks property 3 as the last control flow of non empty element paths is not shared by all of them. Property 2 is broken by the row corresponding to the source activity e_1 where the first control flow of the non empty row elements is not shared by all of them.

	e_1	a_2	e_3	a_4	a_5	a_6	a_7	a_8	a_9
e_1		$\langle\langle e_1, a_2 \rangle\rangle$				$\langle\langle e_1, g_1 \rangle, (g_1, a_6) \rangle$	$\langle\langle e_1, g_1 \rangle, (g_1, g_4) \rangle$	$\langle\langle e_1, g_1 \rangle, (g_1, g_4), (g_4, a_7) \rangle$	
a_2			$\langle\langle a_2, e_3 \rangle\rangle$						
e_3				$\langle\langle e_3, a_4 \rangle\rangle$					
a_4					$\langle\langle a_4, \gamma_3 \rangle, (\gamma_3, a_5) \rangle$				$\langle\langle a_4, \gamma_3 \rangle, (\gamma_3, a_9) \rangle$
a_5									
a_6			$\langle\langle a_6, g_2 \rangle, (g_2, e_3) \rangle$						
a_7			$\langle\langle a_7, g_5 \rangle, (g_5, g_2), (g_2, e_3) \rangle$						
a_8			$\langle\langle a_8, g_5 \rangle, (g_5, g_2), (g_2, e_3) \rangle$						
a_9									

Fig. 5. Matrix M Obtained from the Merge of Fragment Matrices M_{f_1} and M_{f_2} Depicted in Figures 3 and 4

Consequently, the resulting matrix must be rectified to respect the fragment matrix properties. To this aim, we define the following rectification rules:

- When a column depicts multiple non empty elements whose gateway paths do not share the last control flow, then a new convergence configurable gateway is created and inserted between the two objects involved in the last control flow of each non empty element in that column. This is demonstrated in algorithm 1. The algorithm takes as inputs an activity whose corresponding column must be corrected and the merged matrix, and returns the matrix where elements of the input activity corresponding column are corrected. It checks whether there exist elements where the last control flow is not shared by all of them (line 3). If so, then a convergence configurable gateway is created (line 4) and inserted between the source object and the target object of the last control flow of each element path (lines 5-7). This is performed using the function *insertG* which takes as inputs a path, the index of the control flow to be modified, and the object to insert between the control flow objects. For instance, elements corresponding to the target activity e_3 are corrected by inserting a convergence configurable gateway γ_{e1} between the activities a_2 and e_3 and between the gateway g_2 and the activity e_3 . This is depicted in Figure 6.
- By analogy, when a row depicts multiple non empty elements whose gateway paths do not share the first control flow, then a new divergence configurable

Algorithm 1. Merge Control Flow Paths Converging to a Shared Activity

```

1: function MERGECONVERGINGPATHS(Act a, Matrix M):Matrix
2:   begin
3:     if  $\exists b, c. (b, c \in M.A \wedge P_{M.\mu(b,a).length}(M.\mu(b,a)) \neq P_{M.\mu(c,a).length}(M.\mu(c,a)))$ 
    then
4:       Gateway  $\gamma_c \leftarrow new\ Gateway('conv')$ 
5:       for all pa  $\in \bigcup_{i \in M.A} M.\mu(i,a)$  do
6:         pa  $\leftarrow insertG(pa, pa.length, \gamma_c)$ 
7:       end for
8:     end if
9:     return M
10: end function

```

gateway is created and inserted between the two objects involved in the first control flow of all element paths. This is demonstrated in algorithm 2. The algorithm takes as inputs an activity whose corresponding row must be corrected and the merged matrix, and returns the matrix where elements of the input activity corresponding row are corrected. It checks whether there exist paths where the first control flow is not shared by all of them (line 3), in which case, a convergence configurable gateway is created (line 4) and inserted between the source object and the target object of the first control flow of each element (lines 5-7). This is performed using the function *insertG*. For instance, elements corresponding to the row activity a_4 are corrected by inserting a divergence gateway γ_{d1} between the activity a_4 and the gateway g_3 and between the activity a_4 and the activity a_5 . This is illustrated in Figure 6.

Algorithm 2. Merge Control Flow Paths Diverging from a Shared Activity

```

1: function MERGEDIVERGINGPATHS(Act a, Matrix M):Matrix
2:   begin
3:     if  $\exists b, c. (b, c \in M.A \wedge P_{M.\mu(a,b).length}(M.\mu(a,b)) \neq P_{M.\mu(a,c).length}(M.\mu(a,c)))$ 
    then
4:       Gateway  $\gamma_d \leftarrow new\ Gateway('div')$ 
5:       for all pa  $\in \bigcup_{i \in M.A} M.\mu(i,a)$  do
6:         pa  $\leftarrow insertG(pa, 1, \gamma_d)$ 
7:       end for
8:     end if
9:     return M
10: end function

```

When all rules are applied, the obtained matrix turns out to be an adjacency one and can therefore represent a fragment. For instance, the fragment matrix depicted in Figure 6 corresponds to the fragment depicted in Figure 2.

	e_1	a_2	e_3	a_4	a_5	a_6	a_7	a_8	a_9
e_1		$\{((e_1, \gamma_{d1}), (\gamma_{d1}, a_2))\}$				$\{((e_1, \gamma_{d1}), (\gamma_{d1}, g_1), (g_1, a_6))\}$	$\{((e_1, \gamma_{d1}), (\gamma_{d1}, g_1), (g_1, g_4), (g_4, a_7))\}$	$\{((e_1, \gamma_{d1}), (\gamma_{d1}, g_1), (g_1, g_4), (g_4, a_8))\}$	
a_2			$\{((a_2, \gamma_{c1}), (\gamma_{c1}, e_3))\}$						
e_3				$\{((e_3, a_4))\}$					
a_4					$\{((a_4, \gamma_3), (\gamma_3, a_5))\}$				$\{((a_4, \gamma_3), (\gamma_3, a_9))\}$
a_5									
a_6			$\{((a_6, g_2), (g_2, \gamma_{c1}), (\gamma_{c1}, e_3))\}$						
a_7			$\{((a_7, g_5), (g_5, g_2), (g_2, \gamma_{c1}), (\gamma_{c1}, e_3))\}$						
a_8			$\{((a_8, g_5), (g_5, g_2), (g_2, \gamma_{c1}), (\gamma_{c1}, e_3))\}$						
a_9									

Fig. 6. Matrix M after Applying Correction Rules

4.3 Behavior Annotation

In section 2.2, we have explained that to ensure keeping the behaviors of original fragments, we have to *specify behavioral constraints of control flows involving divergence configurable gateways as source objects*. Behavioral constraints, in our work, consist of control flow passing condition. Each one of those passing conditions must involve the last control flow involving a convergence configurable gateway as a target object, and this for each path involving the first stated control flow. Indeed, counter to the work in [13], we do not need to annotate all the control flows of the obtained matrix to keep the behavior of the input fragments, but only particular ones. This passing condition can be retrieved from the merged matrix by performing a backward walk from elements containing paths originating from a given input fragment (saved into an attribute, *provenance*, that is a path attribute containing the fragment label, the path originates from) where their second control flow involves a *divergence configurable gateway as a source object*, until encountering an element containing paths of the same input fragment (also saved into the attribute *provenance*) where their penultimate control flow involves a *convergence configurable gateway as a target object*. Algorithm 3 presents the manner to retrieve the passing condition of a control flow involving a divergence configurable gateway as a source object. The algorithm needs as inputs the source activity of the element containing the control flow whose passing condition must be specified, the control flow, and the fragment matrix. The algorithm returns the control flow that should necessarily be preceded by the input control flow.

The algorithm navigates the matrix and explores every element of the activity corresponding column until finding the element that contains a control flow path whose ultimate control flow involves a convergence configurable gateway as a source object. The latter control flow path must have the same provenance

Algorithm 3. Retrieving the Passing Condition of Control Flows Involving Diverging Gateways as Source Objects

```

1: function RETRIEVECONDITION(Act a, CFlow c, AdjMatrix M): CFlow
2:   declare boolean test  $\leftarrow$  false, CFlow c', ActSet visitedA  $\leftarrow$   $\emptyset$ , Act a'
3:   begin
4:     while M.A - visitedA  $\neq \emptyset \wedge !test do
5:       a'  $\leftarrow random(M.A - visitedA)$ 
6:       if M. $\mu(a', a)$   $\neq \emptyset$  then
7:         CFPPathSet visitedP  $\leftarrow \emptyset
8:         while M. $\mu(a', a) - visitedP$   $\neq \emptyset \wedge !test$  do
9:           CFPath p  $\leftarrow random(M.\mu(a', a) - visitedP)$ 
10:          if p.provenance = c.provenance then
11:            if  $|p| \neq 1 \wedge \exists g. (g \in source(P_{p.length}(p)) \wedge type(g) = 'conv')$  then
12:              c'  $\leftarrow P_{p.length-1}(p)$ 
13:            else
14:              c'  $\leftarrow retrieveCondition(a', c, M)$ 
15:            end if
16:            test  $\leftarrow true$ 
17:          end if
18:          visitedP  $\leftarrow visitedP \cup \{p\}$ 
19:        end while
20:        test  $\leftarrow false$ 
21:      end if
22:      visitedA  $\leftarrow visitedA \cup \{a'\}$ 
23:    end while
24:    if test then
25:      return c'
26:    else
27:      return  $\emptyset$ 
28:    end if
29:  end
30: end function$$ 
```

as the input control flow. To this aim, the algorithm chooses randomly, using function *random*, a row element that has not yet been explored (lines 4-5). This aims to fix the element to explore by its source activity and target activity. The algorithm, then, looks for all the current element paths to find out the one that originates from the same input fragment as that of the input control flow (lines 6-10). If the ultimate path control flow involves a convergence configurable gateway as a source object, then the algorithm returns the penultimate control flow. Otherwise, it calls the algorithm recursively by sending the current element corresponding row activity, and so on until finding the right control flow (lines 11-15). If it is not the case, then this means that the input control flow does not need specify its passing condition.

5 Experimental Result

The proposed approach has been implemented as a tool which retrieves the fragments from a database, represents them following the model presented in definition 1 and performs the merge to generate a consolidated fragment. The experiments are conducted towards proving the effectiveness of our approach by evaluating (i) the compression of the original fragments, as well as (ii) the scalability of the merge algorithm. To this aim, we conducted the experiments on a fragment library that was reported in [15]. The library consists of a shared collection of fragments. The library comprises more than 500 overlapping fragments involving 3445 different activities including 984 activities that appear in more than one fragment, and 26213 control flows. A fragment contains in average 9.83 activities and 46.8 control flow, at most 106 activities and 326 control flows, and at least 2 activities and 3 control flows. We have pre-processed the fragment set in order to keep only couples of fragments that share at least one activity.

The experiments first part evaluates the compression. Indeed, the size of the obtained fragment model is compared to the global size of the input fragments according to the number of activities and the number of control flows as they both influence the understandability of the fragments. We also report the number of configurable gateways that were newly added to fragments during the merge phase (independently of the alignment phase). Indeed, configurable gateways as well as control flows involving configurable gateways as source objects must be configured by the designer at build time. The activity compression factor is defined as $C_A = (100 / \sum_{i \in F} |f_i.A|) * |f.A|$, where F represents the input fragment set and f the resulting fragment. An activity compression factor $C = 50$ means that the input fragments involve the same set of activities, while an activity compression factor $C = 100$ means that the input fragments are disjoint. Similarly, the control flow compression factor is defined as $C_{CF} = (100 / \sum_{i \in F} |f_i.C_f|) * |f.C_f|$. A control flow compression factor $C_{CF} = 50$ means that the fragments involve the same set of objects and objects are linked to each other in the input fragments similarly. The smaller are the compression factors, the more similar are the fragments. Table 7 represents the results of merging couples of fragments with different sizes. The columns correspond, by order, to the number of activities of the first fragment, the number of activities of the second fragment, the number of activities of the merged fragment, the activity compression factor, and number of control flows of the first fragment, the number of control flows of the second fragment, the number of control flows of the merged fragment, the control flow compression factor, and the number of new configurable gateways, i.e. either convergence or divergence ones. The higher is the activity compression factor, the least similar they are. The table also shows that the activity compression factor has rather the same value than that of the control ow meaning that most of control ows are made of activities (few are made of activity and gateway mix). A high number of newly added gateways means that there are many control ows that share either their source or target activities.

	#act ₁	#act ₂	#act _{merged}	C _A	#CF ₁	#CF ₂	#CF _{merged}	C _{CF}	New Conf.	GW.
Min	5	5	8	80	8	11	15	78,94	3	
Max	274	239	344	67,05	312	326	502	78,68	22	
Avg	37,45	35,51	65,82	90,20	75,65	76,511	126,25	82,97	5,04	

Fig. 7. Size Statistics w.r.t. Activity Compression Factors and Control Flow Compression Factors

We also conducted the experiments to evaluate the scalability of our merge algorithm. The experiments are realized on a laptop core i5 intel processor, 2.27 GHz, 4 GB memory running on Microsoft 7. The results show that our merge algorithm supports fragments with a total of 11309 activities and 16141 control flows (simulated fragments) and merges them in around 1sec. Around 80% of the fragment couples are merged in less than a second. Indeed, table 8 shows that even if we increase the number of activities and control flows, this does not alter considerably the execution duration.

#act ₁	#act ₂	#CF ₁	#CF ₂	Merge Time(sec)
274	239	312	326	0.01
32	26	67	82	≈ 0
65	59	87	88	0.003
117	130	155	169	0.012

Fig. 8. Results of the Merge Execution with Time Focus

6 Related Work

The approach proposed by Shuang Sun et al. [16] proposed some patterns to merge business processes represented in petri nets. Indeed, two processes are first analyzed to identify merge points then choose one out of the patterns to decide on the merge type. However, this approach does not fulfill the behavior keeping as no mechanism is presented to handle it.

Gottschalk et al. [17] have proposed to merge business processes to gain synergy effects. This is achieved by first identifying overlapping portions of business processes. In their work, authors propose to transform event-driven process chains into function graphs, made of functions and arrows, where the focus is put on activities for functions and connectors are represented within arrow labels to link functions to each other. However, the approach does not fix restrictions on the behavior that is resulting from the merge. In addition, their model requires that two functions must be separated by an event which is not always the case in real life. Irrelevant connectors are also removed where they depict a single incoming arrow and a single outgoing arrow while for fragments this may be helpful for the designer to preview other behaviors.

Assy et al. [12] propose an approach to facilitate business process design. The approach consists in retrieving and merging existing business process fragments around a particular activity that is given as input. Activities are drawn in a tree construct that is provided with layers representing the distance between activities. Activities represented on adjacent layers are linked by means

of a single edge describing the connectors that figure between the activities. Redundant activities on different layers are then merged in the layer having the least layer index. Among others, the authors provide rules to merge edges, i.e., more specifically edge connectors, linking similar source and target activities and other rules to merge edges going out of a shared source activity (resp. different source activities) and coming to different target activities (resp. shared target activity). In the last case, split connectors (resp. merge connectors) are added to define choices between possible behaviors. The approach lacks precedence constraints as fragments are melted into a single structure. Therefore, the behavior of the input fragments may be inconsistent after a merge construct. Indeed, the additional connector definition is fully left to designers and then is subject to execution blocking. For instance, when an activity is placed after an added convergence connector that joins paths originating from different processes and that the activity waits for an information that is provided by an activity of one path that is not taken.

La Rosa et al. [13] address the problem of merging business processes into a consolidated one. Their objective is that the produced business process must encompass the behavior of the input process and permits to derive them. Authors propose a matching mechanism to check the similarities between input processes. The similarity mechanism is used to retrieve 'maximum common regions' consisting of common nodes and create a unique version in the merged business process. Individual regions are then glued to that region using either a convergence or a divergence connector. Authors also provide reduction rules to clean up the merged process to remove irrelevant constructs that were added during the merge task. All edges connecting nodes are annotated with their provenance, i.e., from which process they were extracted. While this approach overcomes issues in [12] by annotating edges with their provenance, we do not need all of them annotated. This leads to heavy processes in terms of annotations, while, only particular edges must be annotated to allow deriving the input processes.

7 Conclusion

In this paper, we have presented a systematic approach towards merging business process fragments thus helping designers in designing new business processes. In fact, a compact version of several fragments enhances the understandability compared to separate ones. Moreover, the merge helps designers in building a consolidated version of fragments to optimize their integration into complete business processes. At the same time, we have provided resulting fragments with behavioral constraints so as to keep the behavior of the original fragments consistent.

Fragments are first mapped into fragment adjacency matrices then merged into a single version. This approach is systematic as we provide rules in order to perform the merge. Indeed merging converging control flow paths comes to merge elements of a matrix columns while merging diverging control flow paths comes to merge elements of a matrix rows. The behavior keeping is also ensured using backward walks to find out the dependencies between activities.

As a perspective, we aim to provide a complete algebra to merge fragment adjacency matrices with formally defined operators.

References

1. Leymann, F., Roller, D.: Business Processes in a Web Services World: A Quick Overview of BPEL4WS. IBM Software Group, pp. 2–28 (2002)
2. Kokash, N., Birukou, A., D'Andrea, V.: Web Service Discovery Based on Past User Experience. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 95–107. Springer, Heidelberg (2007)
3. Frakes, W.B., Kang, K.: Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering* 31(7), 529–536 (2005)
4. Schumm, D., Karastoyanova, D., Kopp, O., Leymann, F., Sonntag, M., Strauch, S.: Process Fragment Libraries for Easier and Faster Development of Process-based Applications. *Journal of Systems Integration* 2(1), 39–55 (2011)
5. Seidita, V., Cossentino, M., Gaglio, S.: A Repository of Fragments for Agent System Design. In: WOA (2006)
6. Schumm, D., Leymann, F., Ma, Z., Scheibler, T., Strauch, S.: Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls. In: Proceedings of the Multikonferenz Wirtschaftsinformatik, MKWI 2010 (2010)
7. Ouyang, C., Dumas, M., Ter Hofstede, A.H.M., Van Der Aalst, W.M.P.: Pattern-Based Translation of BPMN Process Models to BPEL Web Services. *International Journal of Web Services Research (JWSR)* 5(1), 42–62 (2007)
8. Caetano, A., Assis, A., Tribolet, J.M.: Using Business Transactions to Analyse the Consistency of Business Process Models. In: HICSS, pp. 4277–4285 (2012)
9. Object Management Group. Business Process Modeling Notation (BPMN). Version 2.0 (January 2009)
10. Ouyang, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: From BPMN Process Models to BPEL Web Services. In: ICWS, pp. 285–292. IEEE (2006)
11. Eberle, H., Unger, T., Leymann, F.: Process Fragments. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009, Part I. LNCS, vol. 5870, pp. 398–405. Springer, Heidelberg (2009)
12. Assy, N., Chan, N.N., Gaaloul, W.: Assisting Business Process Design with Configurable Process Fragments. In: IEEE SCC, pp. 535–542 (2013)
13. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Merging Business Process Models. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 96–113. Springer, Heidelberg (2010)
14. Harary, F.: Graph Theory. Addison-Wesley (1991)
15. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on Demand: Instantaneous Soundness Checking of Industrial Business Process Models. In: Data Knowledge Engineering (2011)
16. Sun, S., Kumar, A., Yen, J.: Merging Workflows: A New Perspective on Connecting Business Processes. In: Decision Support Systems, pp. 844–858 (2006)
17. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H.: Merging Event-Driven Process Chains. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 418–426. Springer, Heidelberg (2008)

Provenance-Based Quality Assessment and Inference in Data-Centric Workflow Executions

Clément Caron^{1,2}, Bernd Amann², Camelia Constantin², Patrick Giroux¹,
and André Santanchè³

¹ Airbus Defence and Space, Val-de-Reuil, France

² LIP6 - Univ. de Pierre et Marie Curie (UPMC), Paris, France

³ Univ. de Campinas (UNICAMP), São Paulo, Brasil

Abstract. In this article we present a rule-based quality model for data centric workflows. The goal is to build a tool assisting workflow designers and users in annotating, exploring and improving the quality of data produced by complex media mining workflow executions. Our approach combines an existing fine-grained provenance generation approach [3] with a new quality assessment model for *annotating* XML fragments with data/application-specific quality values and *inferring* new values from existing annotations and provenance dependencies. We define the formal semantics using an appropriate fixpoint operator and illustrate how it can be implemented using standard Jena inference rules provided by current semantic web infrastructures.

1 Introduction

The exponential growth of structured and unstructured information on the web has led to the development of powerful tools for transforming this information into valuable data and knowledge. The WebLab platform[19] provides an open environment to combine these tools into complex media mining workflows using a Web service oriented architecture. WebLab workflows process and transform rich, heterogeneous and complex contents using services which often are very sensitive to the quality of their input data and easily propagate data errors to their results. By combining these services into workflows, low data quality and errors are propagated, the overall quality of the data can degrade very rapidly. This might have critical consequences in the process efficiency by inducing increased operational cost, user dissatisfaction and less effective decision-making. Similar issues have also been observed in other contexts like business processes, data-warehousing, scientific workflows and many different technologies. Solutions have been proposed in the past [17,6] to overcome them (see related work in Section 2).

Example 1. Figure 1 shows an example of a WebLab video mining workflow. For simplicity, XML element types are shown as white rectangles that are consumed and produced by Web services (dark rectangles). Service inputs and outputs are shown by arrows (* signifies multiple occurrences of elements of a given data

type). The workflow is applied to an XML document containing an element of type *video* from which service DEMULTIPLEXER extracts a set of *image* elements and a single *sound* element. Speech recognition service SPEECHREC transforms the *sound* element into a textual *speech* element. The language extractor LANGUAGEEXTR analyzes the *speech* to identify the *language*. Service TRANSLATOR translates the *speech* from the input language to a predefined output language and produces a text element of type *translation*. The named entity extractor ENTITYEXTR extracts a set of *entity* elements from the translated text. The face recognition service FACERECOGNITION extracts and identifies *face* elements from *image* elements. The OCR service OCRSERVICE extracts textual *ocr* elements from *image* elements, using a dictionary of the *language* previously identified by the language extractor.

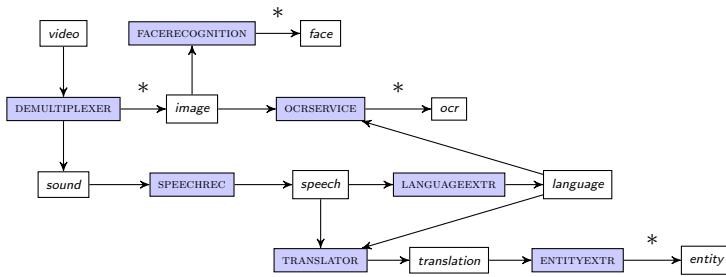


Fig. 1. Media-mining workflow example

Multimedia mining workflows like the one shown in Figure 1 often generate erroneous or low quality data for different reasons. First, the input data itself might be of low quality. Second, some service might not be well adapted or badly configured for transforming some specific input. Moreover, each high-level service (e.g. DEMULTIPLEXER) can itself be materialized by one or several low-level and interchangeable service components and it might sometimes be difficult to choose the right implementation from scratch. Some intermediate data of poor quality might affect the quality of the workflow result. For instance, the extracted *language* might be incorrect when the *speech* element is erroneous. During the quality assessment activity, the expert might find that the results of the entity detection service have low correctness or completeness. Based on *data provenance* information (see Section 3), the system can assist the expert to inspect the service input and observe that the translation itself is not coherent. Further investigations show that the language detected in the *speech* element is not correct. Exploring the results by following this kind of *data and service provenance* paths might be complex and need the participation of different domain experts.

In this article we present a new declarative approach for annotating, deriving and improving the quality of data centric workflows. The goal is to build a quality assessment tool assisting workflow designers and users in exploring and improving the data quality of complex workflow executions, and in particular in

identifying data and services affecting the quality of a given service execution. Users of this tool can explicitly set quality values for intermediate or final data before triggering predefined quality propagation rules to infer the quality of other data nodes. The main contributions are the following:

- We first propose a quality annotation model based on logical expressions over quality values. Our goal is not the formal definition on new quality criteria but to propose an *open* model which allows experts to choose any of the many existing quality criteria [6] that fit their applications. Consequently, the quality annotation process can be complex: some quality values can be generated from the contents itself (e.g. image resolution), others need a more complex analysis comparing existing input data with the generated output (e.g. completeness and precision, correctness). Independently of their complexity, some values can be obtained automatically (e.g. video resolution and colors), others need more human interaction (e.g. syntactic and semantic text consistency).
- Second, we will show how it is possible to assist experts in this annotation process through declarative quality propagation rules inferring new quality values from existing quality annotations. Our quality model is based on a fine-grained provenance generation model [3] including a *declarative provenance mapping* language for specifying service input/output data dependencies.
- Third, we define the formal semantics of our model by using an appropriate fixpoint operator and illustrate how it can be implemented using standard inference mechanisms provided by current semantic web infrastructures.

The paper is organized as follows. In Section 2 we present related work. Section 3 makes a quick review our provenance model. Section 4 presents the concepts and semantics of our quality annotation and inference model. Section 5 describes the current implementation of our quality tool and Section 6 concludes and discusses future work.

2 Related Work

There exists a large literature about the Quality-Of-Service (QoS) assessment and improvement for web service compositions. In this context, quality is important for dynamical selection of web services that satisfy [1] and maximize [25] user defined end-to-end service quality constraints. In this article we focus on Data Quality (DQ) dimensions, independently of QoS criteria such as service availability, response time, or price.

Assessment and improvement of data quality has also been extensively studied in different application contexts. All of the proposed solutions exploit some kind of data provenance information. For example, explicit knowledge about the dependencies between data and their sources are useful to detect copy relationships and to identify reliable and accurate sources that contribute more in deciding the final data values [15]. Provenance information like the creation time, data access services and artifact relationships enables the assessment of the quality

and trustworthiness of Web data [20]. Inconsistencies, conflicts and errors of real-world data that emerge as violations of integrity constraints can be repaired by using dependencies to clean the data [16]. Cleaning uncertain data improves the quality of query answers on probabilistic databases by reducing their ambiguity [12]. The previous approaches are generally restricted to a particular kind of quality criteria (coherency, completeness, precision, trust ...). We propose a more general model where the quality semantics is defined by rules over different quality dimensions. An important contribution in the context of query processing is the provenance semiring model [18,23] which enables the representation of different kinds of data and query provenance information (Trio [7], lineage and why-provenance [9]) as compact polynomial expressions from different commutative semirings. By using specific semirings, the resulting polynomial expressions can be used for estimating/propagating specific kinds of formal quality criteria like trust and data precision. Our approach adopts a different rule-based solution using logical constraints and rules over service parameters.

Provenance plays an important role for assessing data quality in scientific workflows [14] which generate data by a complex analysis of raw data with a mixture of scientific software and human input. In order to assist scientists, scientific workflow systems like Kepler[2], VisTrails[10], Taverna[21] record the process and the raw data that were used to derive the results for tracking down source anomalies, to validate/invalidate intermediate results and to reproduce the data generation process. These systems generally produce coarse-grained provenance information between data and service calls executed as “black-box” system components. Fine-grained provenance links between input and output *fragments* can be captured by “white-box” service models where the precise mapping between the inputs and outputs of a task is known. Fine-grained provenance can be further divided into *where* (which data contributed to the result) and *why* (how did the data contribute to the result) provenance [9,13]. This provenance information can be recorded dynamically during the workflow execution [4] or produced statically by decoupling data and service dependency inference from workflow systems and underlying execution traces [8]. Our provenance generation model [3] adopts the same idea for generating provenance links between XML artifacts using XPath-enabled inference rules. Concerning the exploitation of workflow provenance information for quality estimation, our work can be compared to [24] which also assumes that the quality score for a data set depends on its provenance. The authors present a model and architecture for data quality scoring and apply machine learning techniques to construct a quality function that uses provenance to predict the quality score without the knowledge of other quality related meta-data. Our solution is different in that it is based on explicit quality propagation rules. However, an interesting related open problem is to generate these rules automatically by using an appropriate machine learning techniques.

3 WebLab Model and Provenance

This section shortly presents the WebLab service model and our approach for generating fine-grained provenance links. WebLab workflows generate so-called WebLab documents. Each service call, considered as black-box, receives a WebLab document as input and *extends* it with new resources. This “append” semantics guarantees that no data is ever deleted and the final WebLab documents contain the final workflow result and all intermediate service input and output *resources* (identified XML fragments). Each resource represents any piece of information (document, video, image, text, etc.) that is useful for the final user or for the processing tasks. In this sense, the WebLab model is in line with the notion of nested data collections [5].

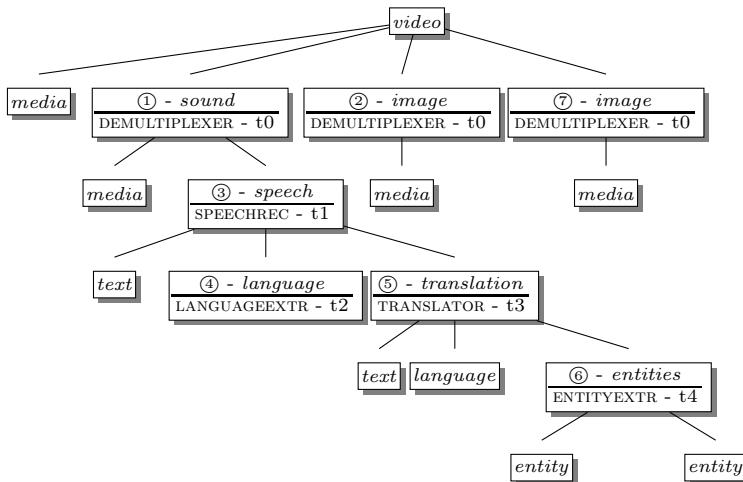


Fig. 2. WebLab document example

The quality inference model proposed in this paper is based on explicit knowledge about which service has generated which document fragments (service provenance) and also which fragments have been used for generating an annotated fragment (data provenance). We proposed in [3] a provenance model for automatically generating data and service dependency links between document nodes. Our provenance model needs to annotate each output XML fragment with a couple (s, t) representing the call of service s at time t that generated it.

Example 2. Figure 2 shows the structure (without contents) of the intermediate XML document obtained after the execution of a workflow branch containing the services DEMULTIPLEXER, SPEECHREC, LANGUAGEEXTR, TRANSLATOR and ENTITYEXTR. The final XML document produced by the workflow will be obtained by the *fusion* of several such intermediate documents generated by the different workflow branches.

The initial document that was given as input to the workflow only contained a *video* element with a *media* child that references external files containing the corresponding multimedia content. All resource nodes (identified XML elements) are labelled by the corresponding calls that produced them. For instance, the XML resource node *speech* whose identifier is ③ is labelled with the service SPEECHREC that generated it at instant t_1 . Service LANGUAGEEXTR is called at time t_2 for identifying the language of the *text* child of the resource node *speech* which is stored in a new resource element *language* whose identifier is ④. The call of TRANSLATOR at instant t_3 translates the same *text* element into some predefined target language and creates a new resource *translation* with identifier ⑤ containing the resulting *text* element as child. Finally, service ENTITYEXTR generates at time t_4 a resource *entities* with identifier ⑥ containing a sequence of *entity* elements.

In order to specify and generate fine-grained provenance links, we apply non-intrusive techniques without modifying the service definition. We define for each service of the workflow *provenance mapping rules* based on *XPath patterns with variables* to specify data dependency links between service's input and output resources (see [3] for details).

Example 3. Provenance rules associated with the services producing the labelled XML sub-fragment in Figure 2 can be defined as following:

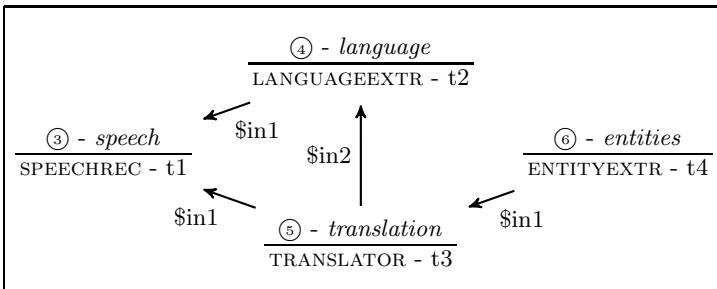
$$\begin{aligned} M_1 &: //speech[$x]/text \Rightarrow //speech[$x]/language \\ M_2 &: //speech[$x]/text, //speech[$x]/language \Rightarrow //speech[$x]/translation \\ M_3 &: //translation[$x]/text \Rightarrow //translation[$x]/entities \end{aligned}$$

Mapping M_1 is applied to service LANGUAGEEXTR and tells that the output resource *language* depends on its sibling *text* element in the same *speech* resource. Mapping M_2 is applied to TRANSLATOR and specify that its result resource *translation* depends both on the element *text* and on the resource *language* which are its siblings in *speech*. Finally, M_3 applied to ENTITYEXTR specify that the extracted entities depend on their sibling *text* in the XML tree.

Mapping rules allow to define the *provenance graph* of some workflow execution as a labelled directed acyclic graph, connecting *each labelled resource* to the other labelled resources that have been used by service calls for its generation.

Example 4. Figure 3(b) shows the provenance graph deduced from the previous mappings and the corresponding Table 3(a) that stores the provenance links. Note that provenance links are defined only between *labelled resources*. The first two columns correspond to the service call (service and timestamp), the third column stores the (input or output) service parameters and the fourth column stores the identifier of the resource node used as parameter. We can see, for example, that resource ④ of type *language* has been produced by service call LANGUAGEEXTR at time instant t_1 and used resource ③ as its input parameter.

service	timestamp	parameter	node
LANGUAGEEXTR	t2	\$in1	(3)
LANGUAGEEXTR	t2	\$out	(4)
TRANSLATOR	t3	\$in1	(3)
TRANSLATOR	t3	\$in2	(4)
TRANSLATOR	t3	\$out	(5)
ENTITYEXTR	t4	\$in1	(5)
ENTITYEXTR	t4	\$out	(6)

(a) Parameter Bindings B (b) Provenance Graph G **Fig. 3.** Provenance Graph and Provenance Table

4 WebLab Quality Model

We present in the following our annotation model for labelling identified XML resource nodes with quality values. Starting from a subset of quality-annotated XML data, quality inference rules allow to infer quality annotations for other unlabelled data by exploiting provenance information.

4.1 Quality Annotation

Our model allows to specify quality labels for different quality criteria which are relevant for the current application. The choice of the specific quality dimensions and the assessment of their values are beyond the scope of this paper.

Example 5. Suppose the WebLab document in Figure 2 produced by a subset of the services of the workflow in Figure 1. Possible quality criteria that are relevant for the media mining task performed by the workflow are shown in Table 1. The *content type* column describes the kind of content that each XML element type corresponds to. Content types are not necessarily XML schema types and they can concretely be represented by a link to some external media file. The values of some quality dimensions can be measured only by inspecting the corresponding content types (eg. image or video resolution). Measurement of other dimensions, such as correctness, completeness or precision can be more complex and need to compare the data to be measured with some other reference values.

Table 1. Element / data quality dimensions

element type	content type	quality dimension	quality values
video	Video	resolution	[460x360, 640x480, 720x480, 800x600, 1024x768]
		colors	integer
		rate	98, 128, 256, ...
image	Image	resolution	[460x360, 640x480, 720x480, 800x600, 1024x768]
		colors	integer
sound	Sound	rate	98, 128, 256, ...
		noise	[0,1]
translation, speech, ocr	Text	consistency	high, low, medium
language	Language	correctness	true, false
face, entity	Entity	correctness	true, false
entities, faces	set(Entity)	completeness	[0,1]
		precision	[0,1]

For example, the **completeness** of the result of the output of ENTITYEXTR can only be measured by looking at the service's input by comparing the number of detected entities with the (estimated or observed) real number of entities in the input.

We define for each quality dimension \mathbf{q} a domain of values, referred to as $dom(\mathbf{q})$. Quality domains can be ordered or unordered. We denote by $\mathbf{q}(n)$ the quality of resource n on dimension q and by $value(q, n) \subseteq dom(\mathbf{q})$ the set of quality values of n for the quality dimension \mathbf{q} . We say that $\mathbf{q}(n)$ is *unknown* iff $value(\mathbf{q}, n) = dom(\mathbf{q})$ and *incoherent* iff $value(\mathbf{q}, n)$ is empty or not continuous (for ordered domains). For each dimension \mathbf{q} we also define a set of comparison operators $\Phi(\mathbf{q})$ such that $\Phi(\mathbf{q}) = \{\leq, \geq, =\}$ if $dom(\mathbf{q})$ is ordered and $\Phi(\mathbf{q}) = \{=, \neq\}$ if $dom(\mathbf{q})$ is unordered.

Example 6. Consider again the example of quality dimensions in Table 1. One can specify for instance the domain for the quality dimension **resolution** as the ordered sequence of discrete values [460x360, 640x480, 720x480, 800x600, 1024x768], for **completeness** as the unordered set of discrete binary values {yes, no} whereas for **noise** the continuous interval [0, 1]. The **resolution** of an image can be chosen as the subsequence [460x360, 640x480] or [460x360] but not [460x360, 1024x768] (not a continuous subsequence).

For each quality dimension \mathbf{q} we can define quality annotations as predicates that compare the quality of a resource with a constant:

Definition 1 (quality annotation). A quality annotation for a dimension \mathbf{q} and a resource n is an expression $(\mathbf{q}(n) \phi a)$ where $\phi \in \Phi(\mathbf{q})$ is a comparison operator and $a \in \text{dom}(\mathbf{q})$ is a constant in the domain of \mathbf{q} .

Annotations can be encoded in a relational *quality annotation table* $A(\text{resource}, \text{predicate})$ containing for each resource n its quality restrictions as predicates. This representation is close to the one of conditional *c-tables* [22] traditionally used to specify incomplete information, where tuples are labelled with boolean combinations of equality predicates.

Similarly to the definition of the quality values of $\mathbf{q}(n)$, we define $\text{value}(q, n, \lambda)$ for a quality annotation $\lambda(q, n) = (\mathbf{q}(n) \phi a)$ as the maximum subset of $\text{dom}(\mathbf{q})$ that satisfy the corresponding predicate. The actual quality value of resource n on dimension \mathbf{q} with respect to all quality annotations $\lambda(q, n)$ in a set A can be computed as the maximum set of values that satisfy the conjunction of all the corresponding predicates:

Definition 2 (quality values wrt. A). Let A be a set of quality annotations. We denote by $\text{value}(q, n, A) = \cap_{\lambda \in A} \text{value}(q, n, \lambda)$ the quality of n on dimension q with respect to the annotations $\lambda(q, n) \in A$.

By extension, $\text{value}(A)$ denotes the set of all quality values that can be inferred from the annotations in A . We say that A is *coherent* iff all quality values in $\text{values}(A)$ are not empty.

Example 7. Consider for example the document in Figure 2. A possible quality annotation table associated to this document is shown in figure 4(a). This table contains quality annotations for the resource nodes ③, ④, ⑤ and ⑥ on different quality dimensions. For each resource, the corresponding quality values that satisfy the annotations in A are shown in table $\text{value}(A)$. For example, the first line in $\text{value}(A)$ shows that the **consistency** value of ③ is *medium*. This value satisfies the conjunction of the constraints stored on the first two lines of A .

resource	annotation
③	consistency (③) < <i>high</i>
③	consistency (③) > <i>low</i>
④	correct (④) = <i>true</i>
⑤	consistency (⑤) = <i>low</i>
⑥	completeness (⑥) ≤ 0.8
⑥	precision (⑥) < 0.9

(a) Quality annotations A

resource	dimension	value
③	consistency	{ <i>medium</i> }
④	correct	{ <i>true</i> }
⑤	consistency	{ <i>low</i> }
⑥	completeness	[0, 0.8]
⑥	precision	[0, 0.9]

(b) $\text{value}(A)$ **Fig. 4.** Quality Annotations and Values

The quality of the XML nodes can change in time since a user might improve its estimation or the system might automatically infer more precise values (see the inference algorithm below). We assume that users can only add quality annotations (the deletion of annotations is not supported).

Proposition 1. Let $A' \subseteq A$ be two sets of quality annotations. The set of quality values $\text{value}(A)$ corresponding to annotations in A is a subset of $\text{value}(A')$ computed for annotations in A' .

Proof. Since $A' \subseteq A$, $\text{value}(q, n, A) = \cap_{\lambda \in (A - A')} \text{value}(q, n, \lambda) \cap \text{value}(q, n, A')$.

4.2 Quality Rules and Inference

We will show in the following the usage of quality rules to infer and refine quality values for resources. Quality rules are associated to services and specify constraints on the their input and output parameters that are linked by provenance links generated by existing mappings. More precisely, we define quality rules for a service s as :

Definition 3 (quality rules). Let s be a service, $\$In = \{\$in1, \$in2, \dots\}$ and $\$Out = \{\$out1, \$out2, \dots\}$ input and output parameters of s linked by provenance links generated by some provenance mapping M . The set $\text{rules}(s)$ contains quality rules for service s , where each rule $r \in \text{rules}(s)$ is follows one of the two following syntactic rules:

1. $r = q(\$in) \phi q(\$out)$ where $\phi \in \Phi(q)$;
2. $r = \{P \rightarrow Q\}$ where P and Q are sets of quality constraints $q(\$p)\phi a$ where $\phi \in \Phi(q)$ and $a \in \text{dom}(q)$;

Example 8. Consider the provenance rule M_1 in section 3 that generates a provenance link from the output *translation* node (denoted as $\$out$) of the service TRANSLATOR to its input nodes *text* (denoted as $\$in1$) and *language* (denoted as $\$in2$). The first quality rule $r1.1$ states that the **consistency** of the text to translate is higher or equal to its translation. Quality rule $r1.2$ states that if the language is incorrect, the **consistency** of the translation cannot be better than *low*.

TRANSLATOR

-
- r1.1 : **consistency**($\$in1$) \geq **consistency**($\out)
r1.2 : **correct**($\$in2$) = *false* \rightarrow **consistency**($\$out$) \leq *low*
-

Example 9. We give in the following more examples of quality rules. For each service, the first line shows its mapping rule, whereas the following lines show the quality rules:

ENTITYEXTR

-
- M2 : $//\text{translation}[\$x] \Rightarrow //\text{translation}[\$x]/\text{entities}$
r2.1 : **consistency**($\$in$) = *low* \rightarrow **precision**($\$out$) \leq *medium*
-

If the consistency of the translated text is low, the precision of the obtained entities cannot be better than medium (rule r2.1).

OCRSERVICE

-
- M3 : $\text{video}[\$x]//\text{image}[\$y], \text{video}[\$x]//\text{speech}/\text{lang} \Rightarrow //\text{image}[\$y]/\text{ocr}$
r3.1 : **resolution**($\$in1$) $< 1024x768 \rightarrow$ **precision**($\$out$) \leq *medium*
r3.2 : **correct**($\$in2$) = *false* \rightarrow **consistency**($\$out$) \leq *medium*
r3.3 : $\$in2 \neq an' \rightarrow$ **precision**($\$out$) \leq *medium*
-

If the resolution of the image is lower than 1024x768, the precision of the generated text cannot be high (rule 3.1). The same is true if the detected language is incorrect or different from english (rules r3.2 and r3.3).

DEMULITPLEXER

$$\text{M4} : //\text{video}[\$x] \Rightarrow //\text{video}[\$x]/\text{images}, //\text{video}[\$x]/\text{sounds}$$

$$\text{r4.1 : } \text{resolution}(\$in) \geq \text{resolution}(\$out1/\text{image})$$

$$\text{r4.2 : } \text{rate}(\$in) \geq \text{rate}(\$out2/\text{sound})$$

The resolution of the extracted images and the rate of the extracted sound cannot be higher than the resolution and rate of the input video.

Rule-Based Quality Inference. The starting point of the rule inference algorithm are the quality annotations for input service parameters in A and the corresponding quality values in $\text{values}(A)$. Parameter bindings allow to replace each $\$In$ and $\$Out$ parameters in a rule r and to obtain the corresponding *instantiated rules*. The inference of new values is obtained by applying the instantiated rules on the initial quality values in $\text{values}(A)$.

Definition 4 (instantiated rule). Let $\text{binding}(s, t, B) \subseteq B$ be the set of input/output parameter bindings for service call (s, t) (t is a timestamp) in a binding table B . Let $r \in \text{rules}(s, R)$ be a binding and a rule of s . The instantiated rule $\text{bind}(r, s, t, B)$ is obtained by instantiations of the parameters $\$In$ and $\$Out$ of service s with their values in $\text{bindings}(s, t, B)$.

The set of instantiated rules obtained for all rules in a set R and a set of bindings B is denoted by $\text{BIND}(R, B)$.

Example 10. Consider the set of quality annotations A in Figure 4(a), the binding table B in figure 3(a) and the set of rules R in Example 8. Then, $\text{binding}(\text{TRANSLATOR}, t3, B)$ returns the bindings $\$in1/\textcircled{3}$, $\$in2/\textcircled{4}$ and $\$out1/\textcircled{5}$. $\text{rules}(\text{TRANSLATOR})$ returns rules $r1.1$ and $r1.2$, $\text{bind}(r1.1, \text{TRANSLATOR}, t3, B)$ is the new rule:

$$\text{consistency}(\textcircled{3}) \geq \text{consistency}(\textcircled{5})$$

where $\$in1$ and $\$out$ have been replaced by $\textcircled{3}$ and $\textcircled{5}$ in $r1.1$ given by the binding $\text{binding}(\text{TRANSLATOR}, t3, B)$. Similarly, $\text{bind}(r1.2, \text{TRANSLATOR}, t3, B)$ returns the new rule:

$$\text{correct}(\textcircled{4}) = \text{false} \rightarrow \text{consistency}(\textcircled{5}) \leq \text{low}$$

The instantiated rules for the set $R = \{r1.1, r1.2\}$ with respect to the bindings B is thus $\text{BIND}(R, B) = \{\text{consistency}(\textcircled{3}) \geq \text{consistency}(\textcircled{5}), \text{correct}(\textcircled{4}) = \text{false} \rightarrow \text{consistency}(\textcircled{5}) \leq \text{low}\}$.

The semantics of the result obtained by applying bindings of a set of rules on the the initial quality values corresponding to annotations in A is defined as follows:

Definition 5 (result semantics). The result of a set of quality rules R with respect to a set of quality annotations A and parameter bindings B is the set of (maximal coherent) quality values satisfying all resource annotations in A and the instantiated rules $BIND(R, B)$.

Example 11. For example, let $R = \{r1.1, r1.2\}$ as shown in example 8, A be the set of quality annotations in Figure 4(a) and B be the set of bindings in Figure 3(a). Then the two quality values $value(\text{consistency}, 3, A) = \{\text{medium}\}$ and $value(\text{consistency}, 5, A) = \{\text{low, medium}\}$ satisfy all quality annotations in A and the instantiated rules in $BIND(R, B)$ ($\text{high} \notin value(\text{consistency}, 5, A)$ because of rule $r1.1$, and $r1.2$ is not “applied” since the detected language is correct).

Rule Evaluation: Algorithm 1 computes the set of quality values V from the initial quality values $values(A)$ by applying iteratively a set of quality rules instantiated with the bindings in B .

Algorithm 1: Rule Evaluation

```

input : quality rules  $R$ , quality annotation table  $A$ ,
        parameter bindings  $B$ 
output: quality value table  $V$ 

1  $V = value(A);$ 
2 repeat
3   for service calls  $(s, t)$  do
4      $B' := bindings(s, t, B);$ 
5     for  $P \rightarrow Q \in rules(s)$  do
6        $PV := value(bind(P, B'));$ 
7       if  $coherent(PV, V)$  then
8          $QV := value(bind(Q, B'));$ 
9          $intersect(V, QV);$ 
10    for  $q(\$in)\phi q(\$out) \in rules(s)$  do
11       $X := bind(q(\$in), B');$ 
12       $Y := bind(q(\$out), B');$ 
13       $UV := value(X \phi value(Y, V));$ 
14       $intersect(V, UV);$ 
15       $UV := value(value(X, V) \phi Y);$ 
16       $intersect(V, UV);$ 
17 until  $V$  does not change;

```

Line 1 initializes all quality values V with the values $value(A)$ corresponding to the quality annotations in A (we suppose that A contains for all resources and for all corresponding quality dimensions \mathbf{q} a default annotation $\mathbf{q}(\textcircled{n}) = dom(\mathbf{q})$). The following loop (lines 2-17) stops when the application of all rules over all service calls (s, t) has no more effect on the set of quality values V (fixpoint

semantics). The two types of quality rules are evaluated differently. The first subloop (lines 5-9) adds the values generated by the set of instantiated constraints (annotations) $bind(Q, B')$ to the set V if the the values generated by the instantiated constraints (annotations) $bind(P, B')$ are coherent with the existing values in V . Operation $intersect(V, QV)$ replaces each quality value in V by its intersection with the corresponding value in QV . The second subloop (lines 10-16) replaces both expression $\mathbf{q}(\$in)$ and $\mathbf{q}(\$out)$ by their binding in B' . Expressions $value(X, V)$ and $value(Y, V)$ extract the corresponding quality values in V for the bound resources. These values are then used to build the instantiated constraints (annotations) UV whose values then replace the corresponding quality values in V by using the intersection operator. We will show in section 5 how this algorithm can be implemented using a standard semantic web (RDF) rule inference engine.

Convergence: The previous algorithm modifies the set of quality values V only by replacing existing values by the intersection with some new computed values (lines 9, 14 and 9). By using a standard monotonicity argument it is easy to show that the algorithm converges to a final set (fixpoint) of quality values after a finite number of steps. We also make the conjecture (by analogy with other logical languages like Datalog without negation) that this fixpoint is unique and independent of the rule evaluation order.

5 Implementation

Figure 5 shows the prototype architecture implementing our provenance and quality model. The architecture is composed of three separate layers. The *WebLab Platform* layer is responsible for the workflow definition and execution. The *Service Catalog* contains all service metadata including the service endpoints and signatures. Each workflow execution generates a single XML document stored in an XQuery-enabled WebLab repository. All XML fragments generated by a given service call are identified and the execution trace is stored in an RDF triple-store for future use by the *WebLab PROV* system. The *WebLab PROV* system [11] generates the provenance and quality graphs. The provenance generator represents the heart of the provenance system. It applies the mapping rules to the WebLab documents by combining them with the corresponding execution trace information. The generated data and service dependencies are also stored in the RDF repository. The quality propagation rules are transformed into the Jena inference rules (see below) and executed by the quality generator module to user added or inferred quality values. See below, for more details about the encoding and inference of quality values. The *WePIGE* user interface interacts with the WebLab PROV system to explore WebLab document trees (document browser), add provenance and quality mappings (mapping designer) and explore data dependency and quality information.

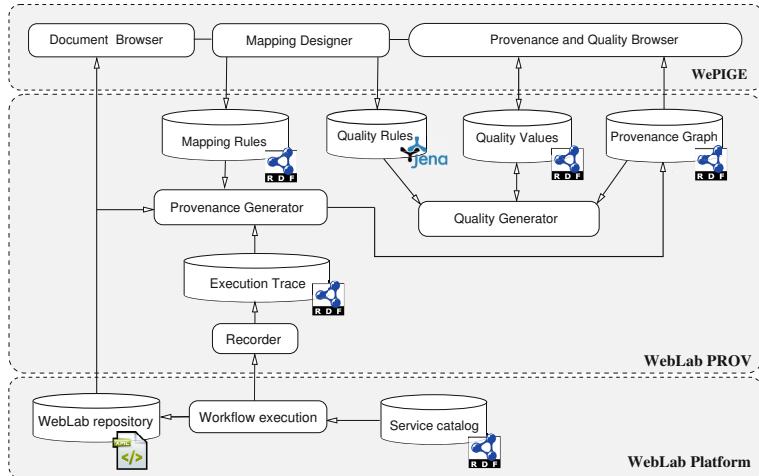


Fig. 5. Prototype architecture

More details about the provenance generation process can be found in [3]. Quality inference is implemented with Apache Jena version 2.11.1 by using an appropriate encoding of quality annotations, values and rules into RDF and Jena rules. We use the Jena generic rule reasoner¹ in forward chaining inference mode (graph saturation): at the creation of a rule and whenever a triple is added or removed to the repository (bound data model), the engine automatically triggers the rules to create or delete new triples in the knowledge base. Similar to SPARQL, Jena Inference rules follow a triple pattern-matching syntax and might use a set of built-in primitives for computing aggregates (*min*, *max*, *product*, *sum*), compare values (*greaterThan*, *lesstThan*), etc. (see the examples below).

The quality model we presented in section 4 defines quality values as subsets or intervals over some quality domain. If the domain is ordered, the corresponding subsets/intervals must be continuous (for example, it is not possible to define a quality value of being either low or high, if there exists a medium value). Based on this constraint and for the sake of simplicity, we encode ordered quality values as finite intervals [*min*, *max*] in some ordered domain (*float*, *integer*) and unordered quality values as values *value* of some unordered domain (*boolean*).

Example 12. Figure 6 shows the encoding of a call of service TRANSLATOR transforming a *speech* element and *language* element into a *translation* element. The **consistency** value of the input SPEECH element is [0.5, 0.8] whereas the LANGUAGE is defined to be correct. Observe also that the **consistency** value of the output TRANSLATION element is [0, 1]

Then for example, quality rules *r1.1* and *r1.2* can be translated into Jena rules as follows.

¹ <https://jena.apache.org/documentation/inference/>

subject	predicate	object
<execution1>	rdf:type	:translator
<execution1>	:timestamp	'1234'^^xsd:integer;
<execution1>	:hasInput1	<speech1>
<execution1>	:hasInput2	<language1>
<execution1>	:hasOutput1	<translation1>
<speech1>	:consistency	<speech-cons1>
<speech-cons1>	:min	'0.5'^^xsd:float;
<speech-cons1>	:max	'0.8'^^xsd:float;
<language1>	:correct	<correct1>
<correct1>	:hasValue	"True"^^xsd:boolean;
<translation1>	:consistency	<translation-cons1>
<translation-cons1>	:min	'0'^^xsd:float;
<translation-cons1>	:max	'1'^^xsd:float;

Fig. 6. Quality data sample**Rule r1.1: Consistency(\$out) \leq Consistency(\$in)**

This rule is translated into two rules for propagating the constraint in both directions (from input to output and vice versa):

```
[translator-consistency:
 (:translator :hasInput1 ?i),
 (:translator :hasOutput1 ?o),
 (?i :consistency ?cons1),
 (?o :consistency ?cons2),
 (?cons1 :max ?value) -> (?cons2 :max ?value)
]
[translator-consistency-inverse:
 (:translator :hasInput1 ?i),
 (:translator :hasOutput1 ?o),
 (?i :consistency ?cons1),
 (?o :consistency ?cons2),
 (?cons2 :min ?value) -> (?cons1 :min ?value)
]
```

Listing 1.1. Consistency(\$in1) \geq Consistency(\$out)

The first rule **translator-consistency** simply binds **?cons1** and **?cons2** to the **consistency** value of the first input *speech* element to the output *translation* element and sets the maximum value of **?cons2** equal to the maximum value of **?cons1** (observe that it is possible but not necessary to delete the old maximal value). Rule **translator-consistency-inverse** symmetrically guarantees that the minimal consistency of the input text is not greater than the minimal consistency of the translation. Obviously the firing of these two rules might generate inconsistent quality values.

Rule r1.2:

```
[translator-language-correctness:
  (:translator :hasInput2 ?i),
  (:translator :hasOutput1 ?o),
  (?i :correct ?correct), (?o :consistency ?cons),
  (?correct :hasValue ?correctValue),
  equal(?correctValue, 'False') -> (?cons :max '0.7')
]
```

Listing 1.2. $\text{correct}(\$in2) = \text{false} \rightarrow \text{consistency}(\$out) \leq 0.7$

Once these rules are defined, the reasoning is done automatically. When applied to the data of figure 6 the quality value (triple) inferred by the previous rules is the triple $\langle <\text{translation-cons1}>, :maximum, '0.8'^\wedge xsd:\text{float}; \rangle$. Observe also that we can simply add this new maximal value which automatically overrides the previous value 1 (without removing the corresponding triple). It is easy to see that this kind of inference might lead to incoherent (empty) quality values which might be automatically detected by defining the corresponding Jena rules (testing $\min > \max$).

6 Conclusion

This paper presents a declarative approach for annotating, deriving and improving the quality of data centric workflows. The proposed method aims to be generic in two directions. First, it can be adapted to different data quality models and any provenance model which allows to compute fine-grained service execution parameter bindings and dependencies. Second, compared to other quality inference models which often deal with specific data quality dimensions (completeness, correctness, trust), with our method users are free to use any kind of quality dimensions over ordered and unordered domains. The annotation model and rule language are designed to be simple and easy to use with appropriate graphical user interfaces. We are currently exploring the different kinds of data quality propagation rules and how their design can be translated into simple graphical user interactions over the workflow schema.

We have also illustrated an implementation of our model using Jena rules and RDF that shows the feasibility of expressing it with standard semantic web technologies. A future work direction is to consider optimization issues by exploring more complex rewritings that reduce processing and storage cost (for example by deleting redundant quality value bounds).

Formal semantics and the rule evaluation algorithm can be further extended by translating our logical quality rules into other formal languages like Petri Nets or Datalog for exploring other formal properties (convergence, coherency) and extensions (stratified negation, disjunction).

Finally, an interesting extension is to materialize the reasoning process in order to be able to obtain the provenance (rules and their application order) of the produced quality values (*i.e* quality provenance). This kind of information would

open new opportunities for analyzing and improving workflows, for example by automatically detecting the “hot spot” services which are strongly influencing the overall workflow results.

References

1. Alrifai, M., Risse, T., Nejdl, W.: A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Trans. Web* 6(2), 1–31 (2012)
2. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S.: Kepler: An extensible system for design and execution of scientific workflows. In: SSDBM, pp. 423–424 (2004)
3. Amann, B., Constantin, C., Caron, C., Giroux, P.: Weblab prov: computing fine-grained provenance links for xml artifacts. In: EDBT/ICDT Workshops, pp. 298–306 (2013)
4. Amsterdamer, Y., Davidson, S.B., Deutch, D., Milo, T., Stoyanovich, J., Tannen, V.: Putting lipstick on pig: enabling database-style workflow provenance. *Proc. VLDB Endow.* 5(4) (December 2011)
5. Anand, M.K., Bowers, S., McPhillips, T., Ludäscher, B.: Efficient provenance storage over nested data collections. In: EDBT, pp. 958–969 (2009)
6. Batini, C., Scannapieco, M.: Data quality: concepts, methodologies and techniques. *Data-centric systems and applications*. Springer (2006)
7. Benjelloun, O., Sarma, A.D., Halevy, A., Widom, J.: Uldbs: Databases with uncertainty and lineage. In: VLDB (2006)
8. Bowers, S., McPhillips, T., Ludäscher, B.: Declarative rules for inferring fine-grained data provenance from scientific workflow execution traces. In: Groth, P., Frew, J. (eds.) IPAW 2012. LNCS, vol. 7525, pp. 82–96. Springer, Heidelberg (2012)
9. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 316–330. Springer, Heidelberg (2000)
10. Callahan, S.P., Freire, J., Santos, E., Scheidegger, C.E., Silva, C.T., Vo, H.T.: Vistrails: visualization meets data management. In: SIGMOD, pp. 745–747 (2006)
11. Caron, C., Amann, B., Constantin, C., Giroux, P.: Wepige: The weblab provenance information generator and explorer. In: EDBT, pp. 664–667 (2014)
12. Cheng, R., Chen, J., Xie, X.: Cleaning uncertain data with quality guarantees. *VLDB Endow.* 1(1), 722–735 (2008)
13. Cui, Y., Widom, J.: Lineage tracing for general data warehouse transformations. *The VLDB Journal* 12(1), 41–58 (2003)
14. Davidson, S.B., Freire, J.: Provenance and scientific workflows: Challenges and opportunities. In: SIGMOD, pp. 1345–1350 (2008)
15. Dong, X.L., Berti-Equille, L., Srivastava, D.: Integrating conflicting data: The role of source dependence. *Proc. VLDB Endow.* 2(1), 550–561 (2009)
16. Fan, W.: Dependencies revisited for improving data quality. In: PODS, pp. 159–170 (2008)
17. Fan, W., Geerts, F.: Foundations of Data Quality Management. Morgan & Claypool Publishers (2012)
18. Foster, J.N., Green, T.J., Tannen, V.: Annotated xml: Queries and provenance. In: PODS, pp. 271–280 (2008)
19. Giroux, P., Brunessaux, S., Brunessaux, S., Doucy, J., Dupont, G., Grilhères, B., Mombrun, Y., Saval, A.: Weblab: An integration infrastructure to ease the development of multimedia processing applications. In: Int. Conf. ICSSEA (2008)

20. Hartig, O., Zhao, J.: Using web data provenance for quality assessment. In: Proc. of the Workshop on Semantic Web and Provenance Management at ISWC (2009)
21. Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. *Nucleic Acids Research* 34(Web Server issue), W729–W732 (2006)
22. Imielinski, T., Lipski Jr., W.: Incomplete information in relational databases. *J. ACM* 31(4), 761–791 (1984)
23. Karvounarakis, G., Green, T.J., Ives, Z.G., Tannen, V.: Collaborative data sharing via update exchange and provenance. *ACM Trans. Database Syst.* 38(3), 19:1–19:42 (2013)
24. Simmhan, Y., Plale, B.: Using provenance for personalized quality ranking of scientific datasets. *I. J. Comput. Appl.* 18(3), 180–195 (2011)
25. Yu, T., Zhang, Y., Lin, K.J.: Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web* 1 (May 2007)

Collaborative Building of an Ontology of Key Performance Indicators

Claudia Diamantini, Laura Genga, Domenico Potena, and Emanuele Storti

Dipartimento di Ingegneria dell'Informazione

Università Politecnica delle Marche

via Brecce Bianche, 60131 Ancona, Italy

{c.diamantini, l.genga, d.potena, e.storti}@univpm.it

Abstract. In the present paper we propose a logic model for the representation of Key Performance Indicators (KPIs) that supports the construction of a valid reference model (or KPI ontology) by enabling the integration of definitions proposed by different engineers in a minimal and consistent system. In detail, the contribution of the paper is as follows: (i) we combine the descriptive semantics of KPIs with a logical representation of the formula used to calculate a KPI, allowing to make the algebraic relationships among indicators explicit; (ii) we discuss how this representation enables reasoning over KPI formulas to check equivalence of KPIs and overall consistency of the set of indicators, and present an empirical study on the efficiency of the reasoning; (iii) we present a prototype implementing the approach to collaboratively manage a shared ontology of KPI definitions.

1 Introduction

Modern networked business paradigms require models offering a common ground for interoperability and collaboration. Reference models have been established for supply chains and other networked organizations [1,2], that include the description of business processes structure and performances as described in a library of Key Performance Indicators (KPIs). These models are typically built as a “consensus view” over the partners’ own views, which is reached mostly by human interaction. In the present paper we propose a logic model for the representation of Key Performance Indicators (KPIs) that supports the construction of a valid reference model (or KPI ontology) by enabling the integration of definitions proposed by different engineers in a minimal and consistent system. The approach is characterised by the representation of what we call the *compositional* semantics of KPIs, i.e. the meaning of KPIs is composed from the meaning of their subparts. As a matter of fact, KPIs are measures generated by means of operations like aggregation and algebraic composition. The aggregative structure of KPIs is captured by the multidimensional model [3], whose formalization is well studied in the Literature. The composite structure of KPIs refers to the *calculation formula* by which an indicator is calculated as a function of other

indicators, for instance in the form of a ratio (e.g. the acceptance rate, or the ratio between income and investments, known as Return On Investment or ROI). The starting point of the paper is that the composite structure of an indicator is at least as fundamental as aggregation to capture its semantics, and deserve special attention.

To illustrate and motivate the approach, let us consider a scenario where a shared dictionary of KPIs is collaboratively managed. Knowledge Engineers and Performance Managers belonging to networked organizations decide to collect KPI definitions for future reference use or for interoperability purposes. Each autonomous entity proposes KPI terms, definitions and properties, then a consensus is achieved in order to decide the dictionary entries. Some typical use cases can be recognised in this process:

- Check of KPIs' identity: this activity allows to establish if a KPI at hand already exists in the dictionary. Two KPIs can be defined to be identical if they provide the same value for the same real phenomenon, expressed by some transactional data. Then, although a “sameAs” relationship can be established by analysing terms and definitions, the ultimate semantics of the indicator is given by its calculation formula.
- Introduction of a new KPI: this is the basic activity allowing the incremental building of the dictionary. Inserting a new KPI should not introduce inconsistencies in the dictionary, hence this activity is naturally related to consistency checking and enforcement mechanisms. For instance, trivial inconsistency can be generated when introducing the income defined as the ratio between ROI and investments, when the ROI definition, as given before, already exists in the ontology.
- Update and deletion of existing KPIs: as the result of coordination and cooperation, KPIs introduced by a single entity can be revised or even deleted from the dictionary. Update and deletion require proper mechanisms for inconsistency management as above.
- Browsing: it is the fundamental activity for the analysis of KPI properties.
- Searching: a dictionary is typically explored to look up KPI definitions. Smart search engines allow to quickly check the existence of a KPI. Besides traditional keyword-based search, the structure of the formula can be used as a key for searching.
- Consensus management and versioning: it refers to the organization of collaborative work, in particular mechanisms for the convergence towards a common uniform view and for tracing and roll-back of previous work.

In this scenario, the proposal of the present paper is to provide advanced support for the above mentioned use cases by leveraging on the representation and manipulation of the formula defining a KPI. In particular, the original contributions of the paper are:

- we combine the descriptive semantics of KPIs with their compositional semantics into an ontology called KPIOnto. The ontology introduces a logical representation of KPIs and their formulas, allowing to make the algebraic relationships among indicators explicit;

- we introduce a framework for reasoning over KPI formulas. The theory includes a set of facts describing KPIs as well as a set of rules formalizing the basic mathematical axioms for equation resolution and formula manipulation. On top of these, a set of rules implements high-level semantic reasoning for KPI identity and equivalence checking, ontology consistency checking, formula and dependencies inference. Experiments on execution time demonstrate the feasibility of the approach;
- through a prototype we also show how semantic reasoning supports the flexible management of a shared ontology of KPI definitions.

The rest of the paper is organised as follows: in the next Section we briefly review the relevant Literature. Sections 3 and 4 introduce the model and related reasoning services respectively. Section 5 is devoted to the evaluation of the approach. Section 6 provides some concluding remarks and discusses future work.

2 Related Work

As observed in [4,5], the development of enterprise models is of particular importance for collaborative networked organizations, as the basis to align people, business processes and technology capabilities, and for the development of methods and tools for better decision-making. The Supply Chain Operations Reference model (SCOR) [1] and the Value Reference Model (VRM)¹ are the two most comprehensive and widely adopted reference models for supply chain management. Both include a dictionary of KPIs, that is a list of KPI definitions, properties and relations with goals and processes. Many other independent dictionaries or libraries were introduced by researcher and international public bodies². These dictionaries and libraries witness the attention toward a systematisation and organisation of the huge amount of existing KPIs, but cannot be considered as KPI models due to their informal nature. Formal models of indicators were recently proposed [6,7,8] in the context of the performance-oriented view of organizations. Description logics and first-order sorted predicate logics are used to express on an axiomatic basis the relations among indicators, by means of predicates like **causing**, **correlated**, **aggregation_of**, however no compositional semantics is taken into account in these models. Furthermore, the models are conceived for the definition of KPIs in a single process-oriented enterprise, and the issue of consistency management is not taken into account. In [9,10] the notion of composite indicator is introduced. Composite indicators are represented in a tree structure, and their calculation with full or partial specification of the formula linking the indicator to its components is also discussed. Related to this work, in [11] the Goal-oriented Requirement Language is enriched with the concept of KPI, and metadata information is added to express the formula. Being in the context of conceptual models for requirement engineering, these papers are concerned with languages and methods for elicitation of specific requirements and do not deal

¹ <http://www.value-chain.org/en/cms/1960>

² See <http://kpilibrary.com/> for a comprehensive list of KPIs and related sources

with general reference models and their consistency. Moreover, formula representation does not rely on logic-based languages, hence reasoning is limited to ad-hoc modules for direct formula evaluation, and no inference mechanism and formula manipulation is enabled.

In the data warehouse field, semantic representations were recently proposed, mainly with the aim to reduce the gap between the high-level business view of indicators and the technical view of data cubes, supporting the automatic generation of customised data marts and OLAP analysis. While the ontological representation of the aggregation structure of KPIs is largely studied [12,13,14,15,16,17], the compound nature and consequent dependency among indicators is much less explored. In [18] the definition of formulas is specified by using a proprietary script language. [19] defines an indicator ontology based on MathML Content to explicitly define the formula of an indicator in order to exchange business calculation definitions and allow for their automatic linking to specific data warehouse elements through semantic reasoning. The approach adopted has strong similarities with our previous work [20]. These approaches are mostly targeted at single enterprises. Despite the increasing interest in collaboration and networking there is still a lack of work in addressing data cubes in distributed and collaborative contexts. A proposal in this respect is [21], where interoperability of heterogeneous and autonomous data warehouses is taken into account. Again, although a preliminary attempt to include formulas in the framework, the compositional semantics of KPIs is missed.

Finally, regarding ontology building, several contributions highlighted the importance of collaborative work, as revealed by the development of dedicated methodologies and tools [22,23]. In this respect, a well-known issue is consistency management during collaborative ontology building. Indeed, the more users concurrently edit the ontology, the more it is likely that a change makes the shared ontology inconsistent. Hence, proper mechanisms to solve inconsistencies need to be defined, to choose which changes to accept. In many cases, such a verification is mainly left to communication and coordination among users; anyway, some work exists which intends to provide a more effective support for such a task. For example in [24], the authors introduce *OSHOQP(D)*, a language of the DL family for modular ontology building and describe an editor based on common wiki systems for the development of modular ontologies represented in such a language. In [25], the authors propose the *SCOOP* platform for the collaborative building of general-purpose knowledge bases. The platform checks for redundancies or conflicts generated by new axioms, providing mechanisms of conflict resolution. These platforms provide support services that are similar to the one proposed in this paper for KPI ontologies.

3 KPIOnto: An Ontology of Key Performance Indicators

From the analysis of dictionaries cited in the previous Section and the multidimensional model of data warehouse domain, we derived the main properties for the development of KPIOnto, an ontology devoted to formally describe indicators.

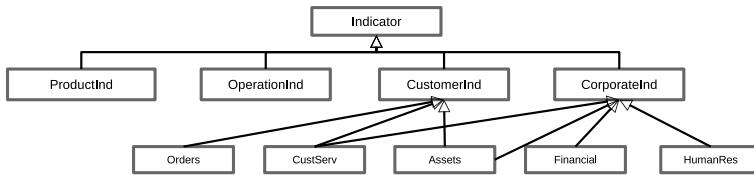


Fig. 1. KPIOnto: a fragment of the Indicator taxonomy

The core of the ontology is composed by a set of disjoint classes, that are detailed in the following: **Indicator**, **Dimension** and **Formula**.

Indicator. It is the key class of KPIOnto, and its instances (i.e., indicators) describe the metrics enabling performance monitoring. Properties of an indicator include name, identifier, acronym, definition (i.e., a plain text giving a detailed description of its meaning and usage), compatible dimensions (see next paragraph), formula, unit of measurement chosen for the indicator (i.e., both the symbol and the description, given by referring to the Measurement Units Ontology³), business object and aggregation function:

$$\begin{aligned}
 \text{Indicator} \equiv & \forall \text{ hasDimension}. \text{Dimension} \sqcap \\
 & \forall \text{ hasFormula}. \text{Formula} \sqcap (=1 \text{ hasFormula}) \sqcap \\
 & \forall \text{ hasUnitOfMeasure}. \text{UoM} \sqcap (=1 \text{ hasUnitOfMeasure}) \sqcap \\
 & \forall \text{ hasBusObj}. \text{BusinessObject} \sqcap (=1 \text{ hasBusObj}) \sqcap \\
 & \forall \text{ hasAggrFunction}. \text{AggrFun} \sqcap (=1 \text{ hasAggrFunction})
 \end{aligned}$$

Following the VRM model, in KPIOnto a taxonomy of KPIs (see Figure 1) is established. However, multiple-categorization is allowed by the ontology, thus supporting more efficient indexing and searching functionalities. As an example, Figure 2 shows the main properties of the KPI *PersonnelTrainingCosts*.

Dimension. A dimension is the coordinate/perspective along which a metric is computed. Following the multidimensional model, a dimension is usually structured into a hierarchy of levels, where each level represents a different way of grouping elements of the same dimension [3]. For instance, Organization can be grouped by virtual enterprise, enterprise, department, business unit, project team and person, while the Time dimension can be arranged in years, semesters, quarters, months and weeks. Each level is instantiated in a set of elements known as members of the level, e.g. the company “ACME” or the month “2013-01”. In order to describe dimensions in KPIOnto, our approach is similar to the one proposed in [26]. In particular, the top class is *Dimension*, which is composed of a set of subclasses, one for each specific dimension, like Organization Dimension, Time Dimension, and so forth. Levels are represented as disjoint primitive subclasses of the dimension (or the set of dimensions) they belong to, whereas each

³ Measurement Units Ontology: <http://idi.fundacionctic.org/muo/muo-vocab>.

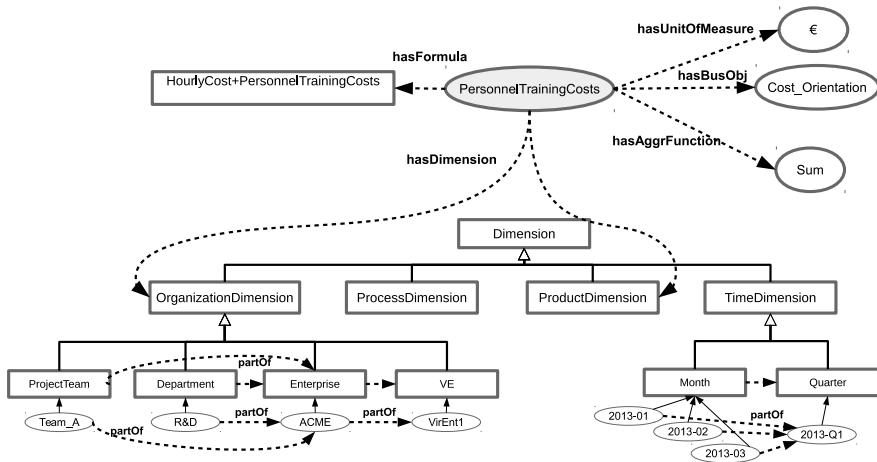


Fig. 2. KPIOnto: properties of indicator PersonnelTrainingCosts

member is an instance of these classes, and the dimension's hierarchy is represented by a transitive partOf^+ relation. An example is shown in the following:

```

VE ⊑ OrganizationDimension
Enterprise ⊑ OrganizationDimension ∩ ∀partOf . VE.VE (=1 partOf)
Department ⊑ OrganizationDimension ∩ ∀partOf . Enterprise.Enterprise (=1 partOf)
VirEnt1:VE, ACME:Enterprise, R&D:Department
partOf(R&D, ACME), partOf(ACME, VirEnt1)

```

from which the reasoner can infer $\text{partOf}(R&D, \text{VirEnt1})$. See also Figure 2, where an excerpt of the hierarchies for Organization and Time dimensions is shown. This ontological description enables usual OLAP operations over KPIs, namely roll-up and drill-down. The roll-up has a direct correspondence with the partOf relationship, while the drill-down implies an opposite path w.r.t. that described by the partOf . Due to the existence of possible branched dimensions (e.g. Person → Department → Enterprise and Person → BusinessUnit → Enterprise), the derivation of this path is not straightforward. In our proposal, the drill-down of a member M at level L_1 is the set of instances of the lower level (L_2), which belongs to the class described by the following expression: $\exists \text{ partof } . M \sqcap L_2$. Referring to the example of Figure 2, we have that $\exists \text{ partof } . \{\text{ACME}\} = \{\text{R&D}, \text{Team_A}\}$, so $\exists \text{ partof } . \{\text{ACME}\} \sqcap \text{Department} = \{\text{R&D}\}$ is the drill-down of ACME along the branch Person → Department → Enterprise.

Formula. A KPI can be either an atomic or a compound datum, built by combining several lower-level indicators. Dependencies of compound KPIs on its building elements are defined by means of algebraic operations, that is a *Formula* capable to express the semantics of an indicator. A formula describes the way

the indicator is computed [27], and is characterised by the aggregation function, the way the formula is presented, the semantics (i.e., the mathematical meaning) of the formula, and references to its components, which are in turn (formulas of) indicators. Given a set $\{f_1, \dots, f_n\}$ of symbols of atomic indicators and a set $\{op_1^{a_1}, \dots, op_n^{a_n}\}$ of operators (in which a_j is the arity of the j -th operator), we define a *well-formed indicator formula* as a finite construction obtained from the recursive definition given below:

- f_j is a well-formed indicator (e.g. NumberOfInternalIdeas or CycleTime);
- if $\{F_1, \dots, F_k\}$ are well-formed indicator formulas then $op^{a_k}(F_1, \dots, F_k)$ is a well-formed indicator formula.

If we refer to algebraic operators like $\{+, -, *, /, ^\}$ then well-formed indicators can be represented by mathematical expressions in prefix notation, like $+(-(A, B), *(C, D))$. Formulas can be graphically represented by a forest of disjoint lattices, where each node is a KPI and its children are the operands of its formula, while leaves are atomic KPIs. The level of a node is defined by 1 + the minimum number of connections required to reach a root node; if more than one root is reachable, then the longest path is considered.

The formal representation and manipulation of the structure of a formula is essential in collaborative networked organizations to check inconsistencies among independent indicator definitions, reconcile indicators values coming from different sources, and provide the necessary flexibility to indicators management.

4 Reasoning-Based Support Functionalities

Reasoning about KPIs is mainly based on the capability to manipulate formulas according to strict mathematical axioms, like commutativity, associativity and distributivity of binary operators, and properties of equality needed to solve equations. Therefore, in order to define KPI reasoning functionalities we represent both KPIOnto formulas and mathematical axioms in a common logic-based layer in Logic Programming (LP). While the descriptive properties of indicators and dimension are represented in OWL2-RL (based on a subset of OWL-DL) that can be directly mapped to LP, indicator formulas are implemented by using two predicates, as follows:

- **formula(Y, Expression, K)** is a LP fact representing the formula related to an instance of the Indicator class in the KPIOnto. Y is the indicator name, **Y=Expression** is the formula of the indicator written using the infix notation, and K is a constant identifying whether Y is an atomic indicator or not. In KPIOnto an indicator is atomic if its formula is not defined on the basis of other indicators.
- **equation(Equation, X)** represents a generic equation in the variable X.

When a new indicator has to be added to the knowledge base, at first it is inserted using the equation predicate, and in this form it goes through correctness and consistency checks, as explained below. If the check is positive a new **formula** fact is added to the LP theory, and hence to the ontology.

In the following subsections we introduce the main functionalities for reasoning about KPIs, grouped by the tasks they are used for: *Formula manipulation*, which include formula rewriting and equation solving (Subsection 4.1); and *Consistency Check*, with functions to check if a new indicator is equivalent to or coherent with others, useful for ontology management (Subsection 4.2).

4.1 Formula Manipulation

Manipulation of mathematical expressions is performed by specific predicates from PRESS (PRLog Equation Solving System) [28], which is a formalization of algebra in Logic Programming for solving symbolic, transcendental and non-differential equations. Its code can be represented as axioms of a first order mathematical theory and the running of the program can be regarded as inference in such a theory. More in detail, given that predicates and functions in PRESS represent relationships among expressions of algebra, it can be interpreted as a Meta-Theory of algebra. The predicates in which it is organised can manipulate an equation to achieve a specific syntactic effect (e.g. to reduce the occurrences of a given variable in an equation) through a set of rewriting rules.

The most interesting, in the context of this work, are those devoted to *Formula Manipulation*, an essential reasoning functionality that consists in walking through the graph of formulas to derive relations among indicators, and rewriting a formula accordingly. We consider two main types of predicates: for the **simplification** and for the **resolution** of equations, on which all the other reasoning functionalities are built. The former are used both to compact and to improve the rendering of a formula, by individuating and collecting equivalent terms (e.g., $a * b$ and $b * a$, or $a * a$ and a^2 , or $a * 1/b$ and a/b), and by deleting unnecessary brackets (e.g. $[] = ()$). The main predicates of this kind are **simplify_term**, which is used to simplify equations, and **simplify_solution** that rewrites the final solution in a more understandable form.

The second type of predicates enables the symbolic resolution of equations by applying mathematical properties (e.g., commutativity, factorization), and properties of equality. The number and kind of manipulations the reasoner is able to perform depend on the mathematical axioms we describe by means of logical predicates. The resolution of the equation **Equation** in a given variable **X** starts from the predicate **solve_equation**, that is defined as follows:

```
solve_equation(Equation,X,X=Solution) :-  
    solve_equation_1(Equation,X,X=SolutionNotSimplified),  
    simplify_solution(SolutionNotSimplified,Solution).
```

where *Solution* is the solution of the equation.

The predicate **solve_equation_1** handles different kinds of equations, e.g. linear and polynomial ones⁴. For lack of space, here we report only the method for solving linear equations, as follows:

⁴ At the present stage of the project, the formulas of all KPIs described in the KPIOnto are linear and polynomial equations, like the majority of indicators available in considered models (e.g., SCOR, VRM, Six-Sigma)

```

solve_equation_1(LeftMember=RightMember,X,Solution) :-
    simplify_term(LeftMember-RightMember,LeftMember1),
    single_occurrence(X,LeftMember1=0),!,
    position(X,LeftMember1=0,[Side|Position]),
    isolate(Position,LeftMember1=0,X=Ssolution).

```

The resolution of linear equations is based on the isolation method, consisting in manipulating the equation and trying to isolate the variable X on the left side of the equation; the right side is the requested solution. To this end the equation is firstly simplified by means of the `simplify_term` predicate, then the `single_occurrence` predicate is verified to check whether the equation is linear or not. The `single_occurrence(X,LeftMem1=0)` predicate is true if the variable X is only in one term of `LeftMember1`. Note that after the simplification, terms of the same degree are grouped together, so if the equation is linear the term of first degree of X is present only once. The other predicates are verified to actually implement the isolation method. The `position` predicate returns the side and list of positions of the variable X in the equation. For instance, the side and position of x in the equation $3*(1/x)-5*y=0$ are respectively “1” (left side) and “1,2,2”, i.e. the first position with respect to the minus operator, the second argument of the product, and finally the second argument of the division. At this point, predicates for isolation are used:

```

isolate([N|Position],Equation,IsolatedEquation) :-
    isolax(N,Equation,Equation1),
    isolate(Position,Equation1,IsolatedEquation).

isolate([],Equation,Equation).

```

The set of predicates `isolax` are needed to move a term from a side to another, for instance multiplying or adding the same term to both sides. In total, in order to perform the formula manipulation functionalities more than 900 predicates are used in the theory.

4.2 Consistency Check

An indicator is consistent with the ontology if three conditions are satisfied: its definition must be unique in the ontology, its formula must not be equivalent to a formula already in KPIOnto and its formula must not contradict any other already defined formulas. The approach we adopt is such that after every insertion of an indicator, the ontology is guaranteed to be consistent. In this way, enterprises always have available an ontology containing valid and usable information. These characteristic is very important considering the domain of the ontology: indicators are used to make decisions, often strategic. In the following, we discuss predicates we introduced to check these conditions.

Identical. The first Consistency Check predicate is introduced to avoid inserting twice the same definition in the ontology. The predicate is defined as follows:

```
identical(Equation,X,X=Solution) :-  
    solve_equation(Equation,X,X=Solution),  
    formula(X,S,_),  
    Solution=S.
```

The reasoner firstly rewrites the `Equation` so that the variable `X` is only on the left side. Then it searches the whole theory for a formula having the same structure of the rewritten `Equation`. If such a formula exists, the predicate is true, and the formula that is identical to the given `Equation` is returned.

Equivalence. During KPI elicitation and ontology population, it is useful to individuate and to manage duplications. In our scenario, two KPIs are duplicated if their formulas are equivalent. If duplicates are found, various policies can be implemented, either by merging the duplicates leaving only one definition for each KPI in the ontology or by allowing multiple definitions by introducing a sort of `sameAs` property among KPIs formulas. The second choice is useful to explicitly represent alternatives, to link a formula to the enterprise that actually uses it, and to reduce the time to make inference.

Given two formulas `F` and `G`, they are equivalent if `F` can be rewritten as `G`, and vice versa, by exploiting Formula Manipulation functionalities. The equivalence check is implemented by means of the following predicate:

```
equivalence(Equation,X,Y) :-  
    expand_equation(Equation,ExpandedEquation),  
    solve_equation(ExpandedEquation,X,X=Solution),  
    formula(Y,S,_), expand_equation(Y=S,Y=ES),  
    solve_equation(Y=ES,Y,Y=Solution2), Solution=Solution2.
```

where `Y` is the formula in the ontology that is equivalent to the `Equation` in the variable `X`.

In order to avoid ambiguity, the check of equivalence is made at level of atomic indicators, where indicators are defined without referring to other indicators. Hence, an equation at this level can not be further expanded. To this end, the `expand_equation` predicate recursively replaces each term in `Equation` with its formula, until we obtain an `ExpandedEquation` that is formed only by atomic indicators. After this transformation the variable `X` could also be at the right side, then the `solve_equation` is called to rewrite the equation. The same is done for each formula in `KPIOnto`. If after these rewritings a formula `Y` exists such that it is equal to the equation, then the `equivalence` predicate is satisfied and the equivalent formula is returned.

Incoherence. When new indicators are added to the ontology or existing indicators are updated or deleted, it is needed to check that such changes do not contradict the ontology. To this end `incoherence` is introduced, that is based on the idea that equivalence relations must be preserved between indicators and formulas. As a consequence, a formula `F1` for a new indicator `I1` is coherent with the ontology if, whenever there exists a formula `F` for an indicator `I` already defined in the `KPIOnto`, such that `F1` can be rewritten as `F`, then the equivalence $I \equiv I_1$ does not contradict the ontology. Otherwise we have an incoherence. In the Prolog theory, we introduced the `incoherence` predicate as follows:

```

incoherence(Equation,X,Y=S) :-
    expand_equation(Equation,ExpandedEquation),
    solve_equation(ExpandedEquation,X,X=Solution),
    formula(Y,S,_), expand_equation(Y=S,Y=ES),
    solve_equation(Y=ES,X,X=Solution2), Solution \= Solution2.

```

Given an equation ($X=Equation$), the predicate returns whether the equation is coherent with the ontology and, in case, the reason for the incoherence. The `incoherence` predicate is based on the definition of equivalence, as described above. Given the set of formulas in the ontology that can be expanded and rewritten as functions of X , each element that is different from the given `Equation` (after the appropriate expansion) leads to incoherence and is returned. We have an incoherence also when, after a manipulation, the new equation assumes the form $0=0$, i.e. the following clause is true:

```

incoherence(Equation,X,inconsistent) :-
    expand_equation(Equation,ExpandedEquation),
    solve_equation(ExpandedEquation,X,0=0).

```

Finally, when the new equation cannot be expanded but a non-atomic formula for X exists, then the new equation contradicts the definition of such a formula. In other cases the coherence is verified.

```

incoherence(Equation,X,X=S) :-
    \+expand_equation(Equation,ExpandedEquation),
    formula(X,S,branch_node), !.

```

5 Evaluation

The goal of this Section is to provide some evidence on the efficiency and effectiveness of the proposed approach. Efficiency is evaluated by reporting the execution times of the main predicates on a real case study. Effectiveness is demonstrated by discussing how the use cases of the collaborative maintenance scenario depicted in Section 1 are supported by a prototype using these predicates.

5.1 Performance Evaluation

The goal of this Subsection is to evaluate the performances of the logical theory implemented in Prolog. To this end, we measured the execution time of each predicate over the real-world ontology produced in BIVEE, an EU-funded project for the development of a platform to enable Business Innovation in Virtual Enterprise Environments⁵.

This ontology takes into account 356 KPIs for monitoring both production and innovation activities, and it has been tested on two Virtual Enterprises in which two end-user partners are involved in. The ontology is characterised by

⁵ <http://www.bivee.eu/>

a lattice of formulas with 281 connected nodes and 75 disconnected ones; the latter represent indicators that are part of the ontology, but are not used to compute other indicators. Indeed the ontology is a set of lattices with different levels (from 1 to 5) and operands (from 2 to 4). On average, each lattice has 3.14 levels and there are 2.67 operands per indicator.

With respect to the predicates described in Section 4, the experimentation focused only on those involving the most complex operations: `identical`, `equivalence` and `incoherent`, that are used for the consistency check of a formula. The others were left out of the test; in fact `solve_equation` and `simplify_*` predicates are internally executed by those previously mentioned.

Test was carried on by evaluating the execution time of each predicate for each connected node of the graph. Note that we discarded the 75 disconnected nodes for which no formula manipulation is needed, hence avoiding an underestimation of execution times. In order to reduce variability in the computation due to the operating system processes that could be running at the same time on the machine, each predicate was executed 10 times and the related running times were averaged. The average time of each predicate was further averaged over all nodes in the graph. Hence, in the following we return one average value for each predicate with the related standard deviation in brackets. In particular for each `formula(X,Exp,K)` in the ontology we evaluated the following predicates: `identical(X=Exp,X,S)`, `equivalence(newPI=Exp+1,newPI,Y)`, and `incoherence(newPI=Exp,newPI,S)`; where `newPI` is a new label which does not exist in the ontology. The introduction of the new label implies that the formula `newPI=Exp+1` can not be equivalent to any formula in the ontology. Similarly, the formula `newPI=Exp` does not introduce inconsistencies. Using these predicates, we ensure that all formulas in the theory were taken into account, hence evaluating the worst situation. Experiments were carried on an Intel Xeon CPU 3.60GHz with 3.50GB memory, running Windows Server 2003 SP 2.

As results, obtained the following execution times: 4 (± 1.1) ms for `identical`, 197 (± 11.5) ms for `equivalence` and 201 (± 10.5) ms for `incoherence`. Results show that `identical` exhibits an almost constant execution time. This outcome is justified by the fact that such a predicate actually performs only a unification, which is independent from the level and number of operands of the node in the lattice. The results of the last two predicates depend on two main components: the time required to find the formula of a KPI in the ontology, and the time needed to evaluate the `expand_equation` predicate; the execution time of other predicates are irrelevant. The former component depends on the number of KPIs in the ontology, hence it is constant in our experiment. The latter component depends on the topology of the lattices, and in particular depends on the characteristics of the sub-lattice of each node, i.e. the level of the node at hand and the number of operands of every node in its sub-lattice. As a matter of facts, let us consider an indicator I at level L whose sub-lattice is formed by nodes with F operands each: the evaluation of equivalence and incoherence for I involves F^L formulas that are needed to evaluate the `expand_equation` predicate. The higher variability of incoherence and equivalence predicates is due to this component. Figure 3

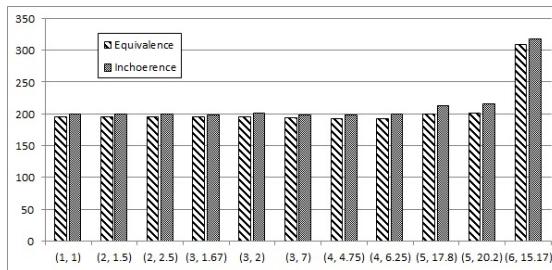


Fig. 3. Average execution time (ms) of the incoherence and equivalence predicates for each pair (level, average number of operands)

reports the dependence of incoherence and equivalence predicates on the topology of the lattice. In particular, the histogram reports for each pair *<level of the node, average number of operands in the sub-lattice of the node>* the average execution times of incoherence and equivalence predicates computed over every KPI in the BIVEE ontology. It is noteworthy that changes in execution times are noticeable only for levels higher than 4.

5.2 The Prototype

To support end-users in the management of KPIOnto in the context of BIVEE project, we developed a web application named *KPIOnto Editor*⁶.

In order to show the support provided by the system, we refer to the following case study in which two enterprises, ACME and ACME2, are collaborating to populate the ontology. Let us suppose that the collaboration has produced an ontology formed by 11 KPIs linked by the following formulas:

- PersonnelCosts=NumHours*HourlyCost*(Overhead+1)
- Costs=TravelCosts+PersonnelCosts
- PersonnelTrainingCosts=HourlyCost*PersonnelTrainingTime
- TeachCosts=PersonnelTrainingTime*HourRate
- InvestmentInEmplDevelopment=PersonnelTrainingCosts+TeachCost

The following subsections describe use cases introduced in Section 1 together with the relative support provided.

Introduction of a new KPI. A new KPI can be introduced only if it is consistent with the ontology, i.e. its formula must not be identical to another formula in the KPIOnto, it must not be equivalent to a formula already in ontology and it must not contradict any other already defined formulas. To this end, the formula of the new KPI is checked by using consistency predicate in this order: **identical**, **equivalence** and **incoherent**. Only if these three are false the new KPI can be introduced.

⁶ <http://boole.dii.univpm.it/kpieditor>

To give an example, let us introduce two new indicators (1) ROI and (2) `TotCostsEmpTrain`, currently not defined, which are aimed to measure respectively the Return On Investment related to new ideas generated by the organization and the total internal costs invested for training of employers:

$$\text{ROI} = \text{ExpectedMarketImpact}/\text{Costs}$$

$$\text{TotCostsEmpTrain} = \text{TeachCosts} + \text{PersonnelTrainingTime} * \text{HourlyCost}$$

Since there are no formulas for `ROI` and `TotCostsEmpTrain` in the ontology, the `identical` predicate returns false and the first step of consistency check is passed. As concerns `equivalence`, the predicate is false for `ROI`, meaning that its formula is not equivalent to any other formula in the ontology. On the contrary, by using the formula about `PersonnelTrainingCosts` and after some formula manipulation, the reasoner discovers that `TotCostsEmpTrain` is structurally equivalent to `InvestmentInEmplDevelopment`. Hence, either the two enterprises can refer to this last one to annotate their data without introducing a new indicator, or a `sameAs` relation linking the two KPIs can be defined. Finally, since `incoherence` for `ROI` returns false, the indicator is added to the ontology together with the atomic indicator `ExpectedMarketImpact` which had not been defined yet in the ontology.

Figure 4 shows the interface used to define a new KPI. The form allows to insert both the descriptive and the structural definition of a KPI. It is built to limit possible syntactic errors, like mistyped dimension's name or a formula referring to undefined KPIs. In the lower part of the Figure, the box is devoted to support the definition of the mathematical formula; the user composes the formula by using a simple equation editor provided with a tool for searching KPIs to include in the formula.

Update of an existing KPI. This functionality is implemented in a way similar to the previous one. The formula of the KPI to update is temporarily removed from the ontology, then the procedure for introducing a new KPI is executed. If the ontology with the updated formula remains consistent then we proceed with the update, otherwise the old formula is restored. For instance, let us consider that an user requires to update `PersonnelTrainingTime`, committing a manifest error when typing the formula (inserting a sum instead of a division): `PersonnelTrainingTime=PersonnelTrainingCosts+HourlyCost`. The new formula directly contradicts the definition of `PersonnelTrainingCost` together with every other higher-level indicators depending on it, i.e. in this case is incoherent with `InvestmentInEmplDevelopment` and the `ROI` added just before. More complex cases can be managed as well, whose discussion is omitted for the lack of space.

The interface implementing this functionality is the same as that presented in Figure 4, but the form is pre-compiled with values from the ontology.

Deletion of a KPI. The procedure used to delete a KPI is more complex: we can remove a KPI only if it does not make the ontology inconsistent. Hence, either the KPI is a root-node, so it is not part of any other formulas, or for each formula in which the KPI appears we can infer at least a new formula which does

The screenshot shows a web-based form for creating a new KPI indicator. At the top, there are several input fields for basic metadata:

- Name:** TotCostEmpTrain
- Acronym:** TCET
- has Bus Obj:** Cost_Orientation
- KPI Description:** Total costs for training of employees
- has UnitOfMeasure:** €
- ProductDimension:** ProductDimension
- OrganizationDimension:** OrganizationDimension
- TimeDimension:** TimeDimension

Below these are dropdown menus for aggregation and formula type:

- has Aggr Function:** Sum
- has Formula:** Atomic Indicator

A sub-form panel is open for defining the formula:

- Name:** F_TCET
- is Additive:** Yes (radio button selected)
- Formula creation:** A numeric keypad interface showing the formula `TeachCost+PersonnelTrainingTime*Hourlycost`.

At the bottom, there is a table titled "Indicators" showing existing entries:

Name	Acronym	Priority Dimension	Unit of Measure	Creator	Action
PersonnelTrainingCosts	PTC	Cost_Orientation	€	D. potena	
PersonnelTrainingTime	PTT	Cost_Orientation	Hours	D. potena	

Buttons at the bottom include "Create", "Reset form", and navigation links for "First", "Previous", "Next", and "Last".

Fig. 4. KPIOnto Editor: form for proposing a new indicator

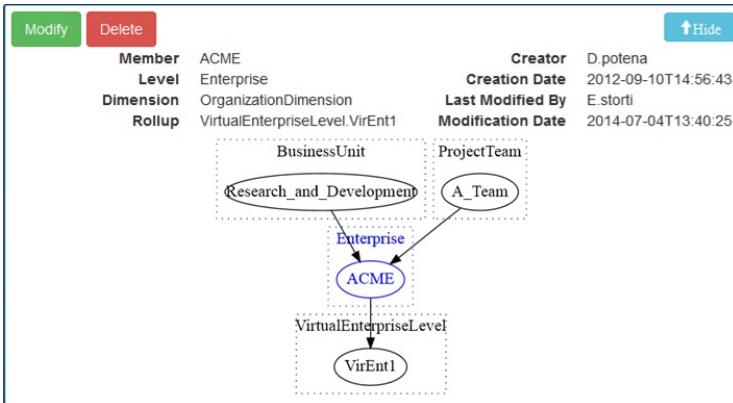
not contains KPI. Since all inferable formulas of an indicator are equivalent, we replace formulas containing the KPI to delete with one of the inferred formulas not containing the KPI. Then, the indicator is removed from the ontology.

The procedure is transparent to the user and does not require a specific interface. In the case the KPI can not be deleted, the system returns formulas that can not be rewritten.

Other functionalities. The *search by term* is simply implemented by querying the descriptive part of the ontology, and the *search by formula* is enabled by unification of the requested pattern with all formulas in the ontology. For instance, searching for the pattern `PersonnelTrainingTime*X`, where X is a variable, the system will return the formulas of `PersonnelTrainingCosts` and `TeachCosts`.

In order to obtain details of a specific indicator, formula or member, the *KPIOnto Editor* provides two panels dynamically generated with information retrieved from the ontology (see Figure 5 and Figure 6). The formula reported in the indicator panel can be interactively browsed: clicking on a KPI, this is replaced with its formula; moving on the name of the KPI, a tooltip with its description is also shown. As for members, together with descriptive information, a graphical representation of the inferred `partOf` hierarchy is provided, as shown in Figure 6. The two buttons in the upper-left corner of both Figures enable the update and deletion functionalities.

		Modify	Delete	↑ Hide
Name	PersonnelTrainingCosts			
Acronym	PTC			
KPI Description	Expenses to train personnel to fulfill tasks related to the project			
Formula Name	F_PTC (is Additive: no)			
	hasFormula:			
	$F_{PTC} = (F_{NTC} * F_{PTT})$			
has Unit Of Measure	€			
has AggrFunction	Sum			
has Dimension	OrganizationDimension			
	ProductDimension			
has Bus Obj	Cost_Orientation			
Creator	D.potena			
Creation Date	2013/9/2T14:14:55			
Last Modified By	D.potena			
Modification Date	2013-09-02T12:14:55			

Fig. 5. KPIOnto Editor: details of a KPI**Fig. 6.** KPIOnto Editor: details of a member

6 Conclusions

In this work we presented a novel model for the formalization of KPI properties, which focuses on characterisation of formulas and reasoning over them. The formal theory allows high-level reasoning supporting most of the activities foreseen in a scenario of collaborative creation and management of a KPI ontology. The feasibility of the approach has been demonstrated by an assessment of the efficiency of reasoning mechanisms on a real case and by the implementation of a prototype enlightening the support for the user. Although reasoning mechanisms ensure that ontology is always consistent, in order to correctly define KPIs, the user needs an adequate knowledge of the structure of the ontology and processes to be measured. Hence the prototype has been designed to be used by Knowledge Engineers, which should be properly trained within the collaborating organizations. At present a Wiki collaborative model is assumed for consensus management and versioning, which is proven to be a powerful model to leverage collective intelligence and effectively integrate diverse points of view in networked environments for unstructured textual contents. A study of the

actual appropriateness of this model in the context of business and performance managers can be in order. Much work can be made to empower and extend the present proposal. Among the most promising directions, there is a tighter integration of descriptive properties of KPIs with their compositional properties, e.g. by combining subsumption and formula manipulation in order to deduce properties of abstract KPI classes. Another interesting direction is the analysis of similarities among formulas, in order to explain the difference among a set of indicators or define similarities classes.

Acknowledgement. This work has been partly funded by the European Commission through the ICT Project BIVEE: Business Innovation in Virtual Enterprise Environment (No. FoF-ICT-2011.7.3-285746).

References

1. Supply Chain Council: Supply chain operations reference model. SCC (2008)
2. Ellmann, S., Eschenbaecher, J.: Collaborative Network Models: Overview and Functional Requirements. In: Virtual Enterprise Integration: Technological and Organizational Perspectives, pp. 102–123. IGI Global (2005)
3. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd edn. John Wiley & Sons, New York (2002)
4. Seifert, M., Eschenbaecher, J.: Predictive performance measurement in virtual organisations. In: Camarinha-Matos, L.M. (ed.) Emerging Solutions for Future Manufacturing Systems. IFIP, vol. 159, pp. 299–306. Springer, Boston (2005)
5. Camarinha-Matos, L.M., Afsarmanesh, H.: A comprehensive modeling framework for collaborative networked organizations. *Journal of Intelligent Manufacturing* 18(5), 529–542 (2007)
6. Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. *Information Systems* 35, 505–527 (2010)
7. del-Río-Ortega, A., Resinas, M., Ruiz-Cortés, A.: Defining process performance indicators: An ontological approach. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 555–572. Springer, Heidelberg (2010)
8. del Río-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortés, A.: On the definition and design-time analysis of process performance indicators. *Information Systems* 38, 470–490 (2013)
9. Barone, D., Jiang, L., Amyot, D., Mylopoulos, J.: Composite indicators for business intelligence. In: Jeusfeld, M., Delcambre, L., Ling, T.-W. (eds.) ER 2011. LNCS, vol. 6998, pp. 448–458. Springer, Heidelberg (2011)
10. Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., Mylopoulos, J.: Strategic business modeling: representation and reasoning. *Software & Systems Modeling* (2012)
11. Pourshahid, A., Richards, G., Amyot, D.: Toward a goal-oriented, business intelligence decision-making framework. In: Babin, G., Stanoevska-Slabeva, K., Kropf, P. (eds.) MCETECH 2011. LNBIP, vol. 78, pp. 100–115. Springer, Heidelberg (2011)
12. Neumayr, B., Schütz, C., Schreftl, M.: Semantic enrichment of OLAP cubes: Multi-dimensional ontologies and their representation in SQL and OWL. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) ODBASE 2013. LNCS, vol. 8185, pp. 624–641. Springer, Heidelberg (2013)

13. Niemi, T., Toivonen, S., Niinimäki, M., Nummenmaa, J.: Ontologies with semantic web/grid in data integration for olap. *International Journal of Semantic Web Information Systems* 3, 25–49 (2007)
14. Huang, S.M., Chou, T.H., Seng, J.L.: Data warehouse enhancement: A semantic cube model approach. *Information Sciences* 177, 2238–2254 (2007)
15. Priebe, T., Pernul, G.: Ontology-Based Integration of OLAP and Information Retrieval. In: Proc. of DEXA Workshops, pp. 610–614 (2003)
16. Lakshmanan, L.V.S., Pei, J., Zhao, Y.: Efficacious data cube exploration by semantic summarization and compression. In: VLDB, pp. 1125–1128 (2003)
17. Nebot, V., Berlanga, R.: Building data warehouses with semantic web data. *Decision Support Systems* 52, 853–868 (2012)
18. Xie, G.T., Yang, Y., Liu, S., Qiu, Z., Pan, Y., Zhou, X.: EIAW: Towards a Business-Friendly Data Warehouse Using Semantic Web Technologies. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 857–870. Springer, Heidelberg (2007)
19. Kehlenbeck, M., Breitner, M.H.: Ontology-based exchange and immediate application of business calculation definitions for online analytical processing. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 298–311. Springer, Heidelberg (2009)
20. Diamantini, C., Potena, D.: Semantic enrichment of strategic datacubes. In: Proc. of the 11th Int. Workshop on Data Warehousing and OLAP, pp. 81–88. ACM (2008)
21. Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., Turricchia, E.: OLAP Query Reformulation in Peer-to-peer Data Warehousing. *Information Systems* 37, 393–411 (2012)
22. Missikoff, M., Navigli, R., Velardi, P.: The usable ontology: An environment for building and assessing a domain ontology. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 39–53. Springer, Heidelberg (2002)
23. Tudorache, T., Noy, N.F., Tu, S., Musen, M.A.: Supporting collaborative ontology development in Protégé. Springer (2008)
24. Bao, J., Honavar, V.: Collaborative ontology building with wiki@nt. a multi-agent based ontology building environment. In: Proceedings of the 3rd International Workshop on Evaluation of Ontology-based Tools, pp. 1–10 (2004)
25. Pease, A., Li, J.: Agent-mediated knowledge engineering collaboration. In: van Elst, L., Dignum, V., Abecker, A. (eds.) AMKM 2003. LNCS (LNAI), vol. 2926, pp. 405–415. Springer, Heidelberg (2004)
26. Neumayr, B., Schrefl, M.: Multi-level conceptual modeling and OWL. In: Heuser, C.A., Pernul, G. (eds.) ER 2009. LNCS, vol. 5833, pp. 189–199. Springer, Heidelberg (2009)
27. Diamantini, C., Potena, D.: Thinking structurally helps business intelligence design. In: D'Atri, A., et al. (eds.) Information Technology and Innovation Trends in Organizations, pp. 109–116. Physica-Verlag HD (2011)
28. Sterling, L., Bundy, A., Byrd, L., O'Keefe, R., Silver, B.: Solving symbolic equations with press. *J. Symb. Comput.* 7, 71–84 (1989)

Aligning Monitoring and Compliance Requirements in Evolving Business Networks

Marco Comuzzi

School of Mathematics, Computer Science and Engineering,
City University London,
Northampton Square, EC1V 0HB London, United Kingdom
marco.comuzzi.1@city.ac.uk

Abstract. Dynamic business networks (BNs) are intrinsically characterised by change. Compliance requirements management, in this context, may become particularly challenging. Partners in the network may join and leave the collaboration dynamically and tasks over which compliance requirements are specified may be consequently delegated to new partners or backsourced by network participants. This paper considers the issue of aligning the compliance requirements in a BN with the monitoring requirements they induce on the BN participants when change (or evolution) occurs. We first provide a conceptual model of BNs and their compliance requirements, introducing the concept of monitoring capabilities induced by compliance requirements. Then, we present a set of mechanisms to ensure consistency between the monitoring and compliance requirements when BNs evolve, e.g. tasks are delegated or backsourced in-house. Eventually, we discuss a prototype implementation of our framework, which also implements a set of metrics to check the status of a BN in respect of compliance monitorability.

1 Introduction

Business processes represent the foundation of all organizations and, as such, are subject to government and industry regulation. Organizations must collect data during business process execution to demonstrate the *compliance* with the regulations they are subject to. Examples of such regulations can be found in different industries and disciplines, such as HIPAA in health care, Basel II and SOX in financial management and accounting, or ISO 9001 for quality management.

Faster market dynamics and fiercer competition also push organizations to focus on their core business, engaging in Internet-enabled, highly dynamic collaborations with external partners, referred to as virtual enterprises or collaborative Business Networks (BNs) [5,4]. BNs are characterized by cross-organizational business processes , i.e. processes that span the boundaries of individual organizations. When (part of) processes are delegated (outsourced) to other partners, organizations may loose visibility over such processes.

This becomes particularly relevant to the case of compliance requirements monitoring. As a result of delegation, compliance requirements will predicate on

tasks executed by several heterogeneous organizations. Collecting the appropriate information to monitor such requirements and keeping this aligned with the compliance requirements defined in the BN may become challenging, since organizations (i) may not want to disclose relevant information, (ii) may not know which pieces of information they should disclose, or (iii) may not be able to capture the required information in their own information systems. For instance, Section 404 of the SOX act states that data security breaches should not be hidden from auditors and always reported to shareholders [14]. A SOX-compliant company A delegating parts of its activities to a company B could remain compliant only if company B would be able to report data security breaches (at least on the data A shared with B to delegate its activities).

Although there are standards, such as ISAE 3402, specifically targeting compliance in service organization outsourcing, these still focus on long-term one-to-one relationships between clients and external service organizations, defining the controls required by clients and the requirements of the service organization to satisfy such controls. Moreover, they take a static perspective on compliance, as they only imply the generation of ex-post reports of periodic audits.

The Internet-enabled BNs that we are considering in this work, however, are intrinsically dynamic, i.e. they are characterized by change, such as partner substitution or process outsourcing exploiting dynamic partner selection [5,2]. In extreme cases, BNs can be *instant* [11], i.e. setup to tackle a specific business need, such as the organization of a large-scale event or the recovery from a natural disaster, and dismantled immediately after the business need has been served. In such a scenario, compliance management cannot rely on long term contracts and ex-post audits, but it should rather mimic the intrinsic dynamicity of the collaboration.

This paper proposes a framework for aligning business process compliance and monitoring requirements in dynamic BNs. Our focus, therefore, is on maintaining the alignment between process compliance and monitoring requirements as BNs *evolve*. In particular, our framework is based on the premise that compliance requirements induce monitoring requirements in the BNs. For instance, to verify the occurrence of a certain task in a process, the organization responsible for the execution of the process needs to provide some evidence of the task execution, such as an event log showing the execution of the task extracted from the company's internal information systems. Given a set of compliance requirements, our framework computes the set of actions required in the BN to maintain the monitorability of the requirements when evolution occurs. Evolution can be at the structural level, e.g. a process or part of it is outsourced from one actor to another actor in the BN, or at the compliance management level, e.g. a new compliance requirement is introduced in the BN.

The steps of the development of our framework are shown in Fig. 1. We first define a conceptual model for representing BNs, their evolution, and their compliance and monitoring requirements. At the conceptual level we also define the mechanisms for aligning compliance and monitoring requirements. The conceptual model is not directly implementable, since it abstracts from specific choices

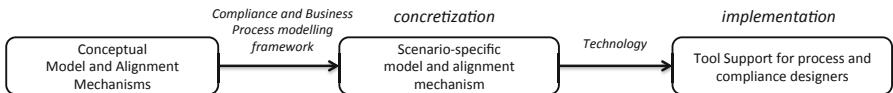


Fig. 1. Steps in framework development

made for the formalization of compliance requirements and for the modelling of business processes. Such choices are taken into account to derive the specific model and alignment mechanisms. We label this step *concretization* in our conceptual framework. Concrete models and mechanisms are implementable into a tool to support the designers in managing compliance and monitoring concerns.

The paper is organised as follows. Section 2 introduces the meta-model underpinning our framework and Section 3 presents the mechanisms to align compliance and monitoring requirements at a conceptual level. Section 4 discusses the operationalisation of our conceptual framework for specific implementation choices regarding the language to express compliance requirements and the chosen framework for business process modelling. A prototype implementing our operationalised framework is presented in Section 5, whereas related work is discussed in Section 6. Eventually, we draw our conclusions and discuss future work in Section 7.

2 Conceptual Model

The conceptual model underpinning our framework is shown in Fig. 2.

In our model, a business network is constituted by a set of business entities (actors), which participate to business processes. Participation has to be intended here at the operational level, that is, actors execute specific parts of a business process to which they participate. As such, actors may be able to provide the information required to monitor compliance of the business processes to which they participate.

Without loss of generality, we consider single-entry, single-exit, block structured processes [19]. Hence, a process is constituted by a set of blocks. Blocks are the process decomposition unit over which we define compliance requirements. Blocks can be outsourced to another actor in the BN. In our model, the outsourcing relationship is captured at the process structural level, that is, an outsourced block points to the process to which it has been outsourced to.

We deal with processes specified using the *public view* [12,16,5]. The public view of a process retains those aspects that are relevant for the collaboration among partners in a BN and hides the internal detailed specification required for process enactment by individual partners (specified in the *private view*). Note that we are not concerned with the control flow of processes. Being able to monitor a set of compliance requirements, in fact, does not depend on the particular path followed by process execution. In other words, we focus on design-

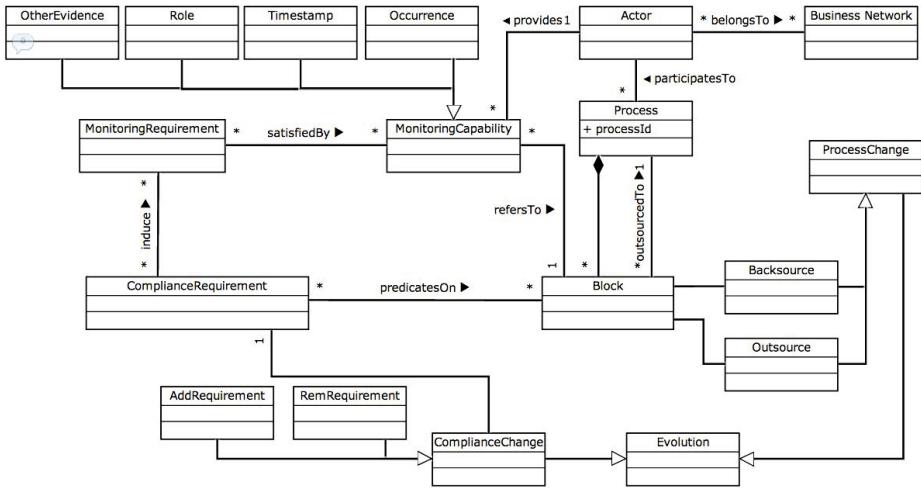


Fig. 2. Conceptual Model of BN and compliance

time compliance, i.e. designing BNs and processes that *can be* monitored [22], irrespectively of the particular path followed by process instances.

As far as compliance is concerned, a BN is characterized by several compliance requirements. A compliance requirement refers to any explicitly stated rule of regulation that prescribes any aspect of business processes in a BN [8,22].

In order to be monitored, compliance requirements induce monitoring requirements, i.e., the information that actors participating to a certain process should provide to guarantee that compliance requirements can be verified. Monitoring requirements are satisfied by actors through the implementation of monitoring capabilities. A monitoring capability is conceptually defined as the ability of an actor to provide the information about a process required to check the satisfaction of one or more compliance requirement.

We identify four types of monitoring capabilities, i.e. Occurrence, Timestamp, Role, and Other Evidence. The Occurrence capability is the ability to demonstrate the execution of a certain activity or block thereof. A Timestamp capability refers to the ability of providing reliable information about the instant in time in which an activity has been executed, e.g. in the form a timestamped event. Timestamp information is required, besides evidence of the occurrence, to monitor compliance requirements referring to the ordering of activities. A Role capability refers to the ability of a business entity to provide reliable information about the role (person, department, business unit) executing a certain activity. This capability is required for instance to monitor segregation of duties. These three capability types capture information about the *provenance* of processes [3], i.e. what manipulation have been performed in the process, when and by whom.

Eventually, the OtherEvidence capability refers to the ability of an actor to provide evidence demonstrating that a certain activity has not occurred. While

non-occurrence cannot be demonstrated unequivocally, such evidence may be provided in the form of a list of all execution traces of a process not showing the execution of the block for which non-occurrence is required [22].

We argue that the above four types of monitoring capabilities can cover most of the monitoring requirements derived from compliance requirements expressed using LTL or other languages that can be translated into LTL, such as Event Calculus. A discussion of how the above types of monitoring capabilities cover the LTL compliance patterns considered by [8] is presented in Section 4.1.

Eventually, as far as evolution is concerned, for reasons of brevity we consider a subset of the BNs evolution types considered by [5] in the context of SLA management in evolving business networks. The evolution of a BN can occur at the level of compliance requirements or processes. New compliance requirements can be introduced (added) or existing compliance requirements may no longer be necessary for a BN (remove). Processes or, more specifically, blocks within processes, can be either outsourced by one partner to another in the BN or backsourced in-house.

3 Compliance-Monitoring Alignment Mechanisms

The mechanisms to align monitoring and compliance requirements can be conceptually defined on the basis of the conceptual model introduced in the previous section. A formal characterization of such mechanisms can only be given once specific implementation choices have been made, such as choosing the language to express compliance requirements or the way outsourcing is captured in business process models. Examples of formal characterizations of our alignment mechanisms for specific implementation choices are presented in Section 4.2.

The mechanism to align compliance and monitoring requirements when a new compliance requirement `new_cr` is introduced is presented in Alg. 1. In a nutshell, for all the blocks on which `new_cr` predicates, the *single alignment check* procedure is called. In such procedure, first we assess whether the block under consideration is outsourced. If that is not the case, then we derive the monitoring requirements induced by `new_cr` and, consequently, the monitoring capabilities required to satisfy such monitoring requirements. Note, in fact, that monitoring capabilities can be implemented only by the actors actually participating to a process on which the new compliance requirement predicates. If the block is outsourced, then the procedure is recursively called on the process to which the block is outsourced. Such recursive call allows the mechanism to execute over arbitrarily long outsourcing *chains*. This is consistent with techniques commonly adopted for the transitive propagation of change in collaborative business processes [5,9].

As a result, the mechanism described by Alg. 1 returns a set of new monitoring capabilities that must be created to guarantee the alignment between monitoring and compliance requirements. Note that some of such monitoring capabilities may have already been created to preserve alignment of previously introduced

```

/* main procedure */  

Get blocks on which new_cr predicates;  

foreach block do  

| run single alignment check of new_cr, block;  

end  

/* single alignment check of new_cr, block */  

if block is not outsourced then  

| Calculate monitoring requirements induced by new_cr on block;  

| Create monitoring capabilities to satisfy identified monitoring  

| requirements;  

else  

| find block in outsourced process;  

| /* recursive call */  

| run single alignment check of new_cr, block;  

end

```

Algorithm 1. Alignment with monitoring requirements of a new compliance requirement new_cr

compliance requirements. The information returned by the alignment mechanism is used by the designer to:

- request the creation of missing monitoring capabilities to the actors who can provide them;
- connect existing monitoring capabilities to new compliance requirements, to guarantee their verification at runtime.

For lack of space, we omit the algorithmic representation of the mechanism to ensure alignment when a compliance requirement is deleted. In principle, a compliance requirement may be deleted without the need for any additional actions. However, we also argue that each actor in a BN should disclose only the minimal amount of information required to monitor compliance. Therefore, when a compliance requirement is deleted, all its required monitoring capabilities may also be deleted. Deleting monitoring capabilities is suggested by our framework only if they are not required by other compliance requirements in the BN.

Alg. 2 presents the mechanism to align monitoring and compliance requirement when a block of a process is outsourced. The mechanism simply runs the *single alignment check* defined by Alg. 1 on all compliance requirements predicated on the block that has been outsourced.

For lack of space, we omit the algorithmic representation of the mechanism to ensure alignment in the case of backsourcing. Similarly to what described for outsourcing, the monitorability of each existing compliance requirement involving a block that is backsourced should be rechecked as new.

```

/* outsourcing alignment mechanism */  

Get list of compliance requirements cr predicating on block;  

foreach cr do  

| run single alignment check of cr, block;  

end

```

Algorithm 2. Alignment of monitoring and compliance requirements in case of outsourcing of a block

4 Concretization for Specific Implementation Choices

As introduced before, in order to capture a real world scenario, a concretization step is required (see Fig.1). Concretization involves the specification of the following aspects:

- the format of compliance and monitoring requirements;
- the framework chosen for modelling business processes (and, specifically, outsourcing and backsourcing).

Once the above aspects are specified, it would be possible for a designer to use our conceptual model and alignment mechanisms to model a BN and manage the alignment of compliance and monitoring requirements as the BN evolves.

In this section we discuss a set of choices for the *concretization* of our conceptual framework. Specifically, the implementation choices described here are the ones implemented by the prototype described in Section 5. This discussion is also provided as a reference example to support designers who may need to accommodate different implementation choices for the concretization of our conceptual model and alignment mechanisms.

We first discuss the derivation of monitoring requirements and capabilities for a specific way of expressing compliance requirements and our choice to model outsourcing based on an extension of BPMN in Section 4.1. Then, in Section 4.2, we derive a concrete implementation of the alignment algorithms presented in Section 2 based on our concretization choices. For illustration purposes, we also discuss an example of an evolving BN in the healthcare industry in Section 4.3.

4.1 Concretization of Monitoring Capabilities and Outsourcing Model

Compliance requirements can be specified using a variety of languages, such as event calculus [21], deontic logic [23], or LTL [8]. Similarly, business process outsourcing in structured processes can be modelled using different notations, such as BPMN, BPEL, or structured EPCs [13].

We consider the set of atomic compliance patterns adopted in [8], which are reported in the first column of Table 1. Patterns involve one or two blocks as operands. Note that arbitrarily complex (composite) compliance requirements

Table 1. Atomic Compliance Requirements Patterns and Required Monitoring Capabilities

Compliance Pattern	Description	Occr(-)	Tmst(-)	Role(-)	OtEv(-)
exists A	A must occur in the BN	A	×	×	×
A precedes B	B must always be preceded by A	A, B	A, B	×	×
A leadsTo B	A must always be followed by B	A, B	A, B	×	×
A segrFrom B	A and B assigned to different roles	×	×	A, B	×
A inclusive B	Presence of A mandates presence of B	A, B	×	×	×
A prerequisite B	Absence of A mandates absence of B	×	×	×	A, B
A exclusive B	Presence of A mandates absence of B (and vice versa)	A, B	×	×	A, B
A substitute B	Presence of A substitute absence of B	A	×	×	B
A corequisite B	Either A and B are present together, or absent together	A, B	×	×	A, B

may be derived by combining the atomic patterns of Table 1 using standard logical operators. As pattern composition does not introduce any specificity to our approach, we will not consider it further in this paper.

Table 1 classifies the monitoring capabilities required by the compliance requirements considered in this work and matches these with the compliant requirements patterns. Given the monitoring requirements, from a technical standpoint a monitoring capability becomes a point of access, i.e. an interface, for interested stakeholders, such as auditors or compliance experts, to access the information implied by the monitoring requirement. For example, in order to check the satisfaction of the compliance requirement **exists A**, a proof of the occurrence of A is required, for instance in the form of an information system log showing the execution of A in at least one process instance. This information has to be made available by the actor executing A, for instance through an operation of a monitoring Web service.

As far as outsourcing is concerned, we extend our conceptual model as depicted in Fig. 3(a). We consider three types of blocks, namely internal, outsourced, and placeholder blocks [5]. Internal blocks are executed internally (*in-house*) by one specific actor. The actor may be able to provide the required monitoring capabilities involving such a block. Placeholders blocks can only be part of outsourced blocks and are introduced to maintain a link between the process structure and the related compliance requirements. Fig. 3 presents also a generic example of business process outsourcing. In Fig. 3(b) the actor **actor1** executes all required tasks in house, i.e. all blocks of the business process are internal. Fig. 3(c) shows the resulting BN when blocks B and C are outsourced to **actor2**. After outsourcing, the process executed by **actor1** includes an outsourced block **BC_OUT** (represented, with an abuse of notation, as a BPMN expanded process in the figure), which will reference the process executed by **actor2**. The outsourced block includes placeholder blocks bearing the same name as the internal blocks that have been outsourced, namely B and C. These are required

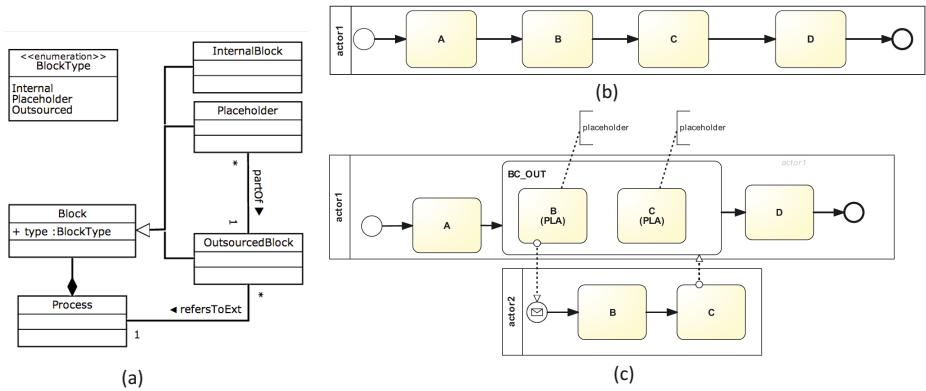


Fig. 3. Outsourcing modelling and generic example

to keep the consistency between compliance requirements and the blocks they predicate on. If, in fact, blocks B and C would be simply removed after outsourcing, then it would be impossible to know that compliance requirements originally predicated on such blocks should now refer to the related internal blocks of the process of actor2.

Note that, for the sake of simplicity, we also make the hypothesis that the internal blocks of the process executed by the outsourcer, i.e. actor2, bear the same name as the internal blocks originally outsourced (and now become placeholders). Generally, string equality may be substituted by any mapping function between the labels chosen for placeholders and the correspondent internal blocks in the referenced outsourced process.

4.2 Concretization of Alignment Mechanisms

In this section we describe the concrete mechanisms implementing the conceptual alignment mechanisms described by Alg. 1 and Alg. 2. The concretization is based on the implementation choices regarding compliance requirements, monitoring capabilities, and outsourcing discussed in the previous section.

Alg. 3 shows the mechanism required to preserve alignment when a new compliance requirement **cr** is added to the BN. The mechanism returns a set of actions required to ensure alignment, i.e. a set of monitoring capabilities **mon-CapSET** that should be implemented. The main procedure **PreserveMonitorability()** calls a sub-procedure **checkMonitorability()** on all operands of **cr** (see line 5). The sub-procedure computes the required monitoring capabilities for each internal block on which **cr** predicates (line 3). If a required monitoring capability is not currently available in the BN, then its creation is added to the list of actions required to ensure monitorability (lines 4-8). If an operand is a placeholder, then the sub-procedure is recursively called on the correspondent internal block of the process to which the placeholder is outsourced (line 13). Such a recursive call allows the functioning of the mechanisms on arbitrary long outsourcing *chains* in the BN.

```

1. procedure:: PreserveMonitorability(cr:ComplianceRequirement)
2. monCapSET:MonitoringCapability[]  $\leftarrow \emptyset$  {create empty list of monitoring
   capabilities}
3. opSET:Block[]  $\leftarrow$  cr.predicatesOn
4. for all blk  $\in$  opSET do
5.   monCapSET  $\leftarrow$  monCapSET  $\cup$  checkMonitorability(cr, blk, monCapSET)
6. end for
7. return monCapSET
8. end procedure

1. procedure:: checkMonitorability(cr:ComplianceRequirement, blk:Block,
   monCapSET:MonitoringCapability[])
2. if blk.type = INT then
3.   mrSET:MonitoringRequirement[]  $\leftarrow$  cr.induce{Get induced monitoring
   requirements}
4.   for all mr  $\in$  mrSET do
5.     if  $\#$  a: MonitoringCapability : mr.satisfiedBy=a  $\wedge$  a.refersTo=blk then
6.       monCapSET  $\leftarrow$  mc:MonitoringCapability: mr.satisfiedBy = mc  $\wedge$ 
          mc.refersTo=blk {Add required monitorig capability to list}
7.     end if
8.   end for
9. end if
10. if blk.type = PLA then
11.   pro:Process  $\leftarrow$  blk.partOf.refersToExt {get process to which block is
    outsourced}
12.   iblk:InternalBlock  $\leftarrow$  b: b.name = blk.name  $\wedge$  b  $\in$  pro {find block in
    outsourced process}
13.   checkMonitorability(cr, iblk, MonCapSet) {recursive call}
14. end if
15. return monCapSET

```

Algorithm 3. Concrete alignment for new compliance requirements *cr*

The concrete mechanism ensuring alignment in case of outsourcing is shown in Alg. 4. It takes as input a set of blocks *BLK* to be outsourced and a reference *extPro* to the external process where these have to be outsourced. The mechanism first checks whether outsourcing is feasible, that is, whether the external process has blocks bearing the same name as the internal blocks to be outsourced (lines 4-6). Note that in this work we are not concerned with the internal semantic equivalence of tasks, but we only consider mapping equivalence among task labels to operate the internal/placeholder block substitution.

Then, the mechanism adjusts the structural aspects of the BN. In particular, the internal block(s) to be outsourced should be encapsulated into an outsourced block and each of them replaced by a placeholder bearing the same name (lines 7-12). A reference from the created outsourced block to the external process

```

1. procedure:: Outsource(BLK:Block[],extPro:Process)
2. isFeasible:boolean  $\leftarrow$  true
3. for all blk  $\in$  BLK do
4.   if  $\nexists$  b:Block s.t. b  $\in$  extPro  $\wedge$  b.name = blk.name then
5.     isFeasible  $\leftarrow$  false; break;
6.   end if
7.   if isFeasible then
8.     TMP: Block[]  $\leftarrow$  BLK {Save blocks to outsource for later...}
9.     newOutBlk:OutsourcedBlock  $\leftarrow$  new OutsourcingBlock()
10.    newOutBlk.refersToExt  $\leftarrow$  extPro {Set ref. to outsourcing process}
11.    for all blk  $\in$  BLK do
12.      blk  $\leftarrow$  new Placeholder(); {Create placeholder in place of internal
        block}
13.      blk.partOf  $\leftarrow$  newOutBlk {set ref. to outsourced block}
14.    end for
15.    monCapSET:MonitoringCapability[]  $\leftarrow$   $\emptyset$ 
16.    for all cr:ComplianceRequirement s.t. cr.predicatesOn  $\subseteq$  TMP do
17.      monCapSET  $\leftarrow$  monCapSET  $\cup$  PreserveMonitorability(cr) {Restore
        monitorability of involved compliance requirements}
18.    end for
19.  else
20.    !! Outsourcing not feasible !!
21.  end if
22. end for

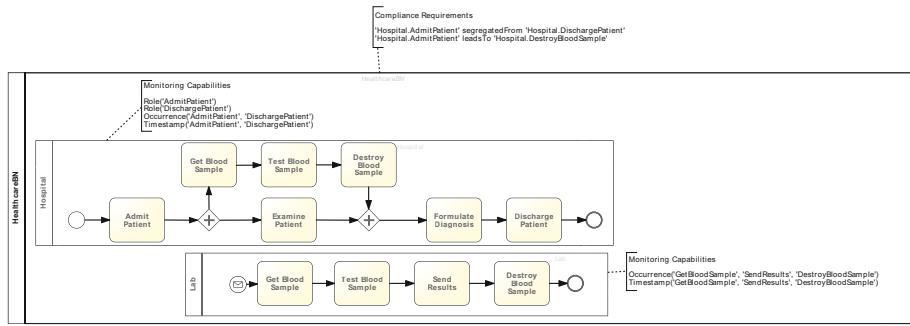
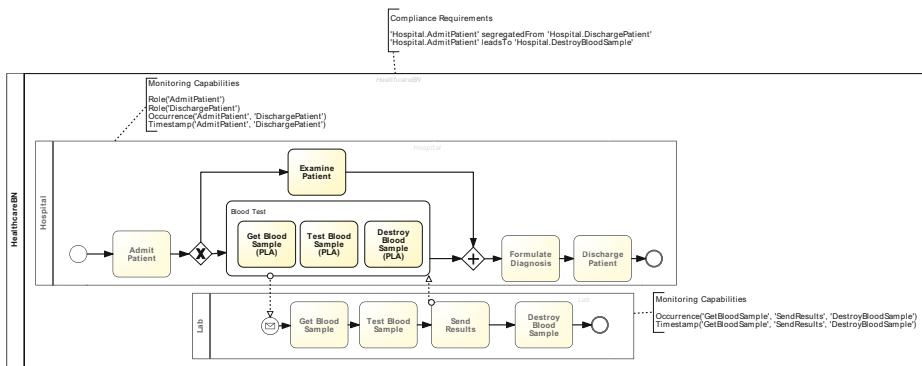
```

Algorithm 4. Concrete alignment in case of outsourcing

should also be set (line 13). Once the structure of business processes in the BN has been updated correctly, a monitorability check should run for all compliance requirements involving the blocks outsourced (lines 15-17). As a result of outsourcing, in fact, the blocks involved by some compliance requirements may have changed from internal to placeholders. This may bear an impact on their monitorability. Therefore, each compliance requirement is now treated as new (hence, the procedure `PreserveMonitorability()` described in Algorithm 3 is called).

4.3 Example

As an example, we consider an adaptation of the healthcare business network considered in [16], depicted in Fig. 4. Initially, the network involves two actors, i.e. Hospital and Lab, and two processes, i.e. Patient Treatment and Blood Test. The two actors are not collaborating at the moment, as there is no relationship between the two processes. Fig. 4 also shows two compliance requirements for the process contributed by the Hospital. The first requirement relates to segregation of duties, that is, for quality reasons the doctor performing the discharge should differ from the doctor who examined the patient. The second requirement

**Fig. 4.** Running Example: Healthcare BN**Fig. 5.** Outsourcing blood testing in the running example

predicates that, for privacy reasons, blood samples should always be destroyed after being tested. Note that all blocks in the processes of Fig. 4 (represented as BPMN tasks) are internal.

Given the monitoring capabilities of the Hospital reported in Fig. 4, only the first compliance requirement can currently be monitored. Monitoring the second requirement requires occurrence and timestamp capabilities over the block `DestroyBloodSample`, which are currently not available for the Hospital.

Fig. 5 shows the BN resulting from the outsourcing by the Hospital of the blood testing to the Lab. Because of outsourcing, the framework recognizes that the task `DestroyBloodSample` for the Hospital becomes a placeholder for a task that has been outsourced to the Lab. The outsourcing changes monitoring requirements of the BN and, therefore, their alignment with the compliance requirements. In this specific case, the compliance requirements are now aligned, as the monitoring capabilities required to monitor the compliance requirements that could not be provided by the Hospital before outsourcing (see Fig. 4), can now be provided to interested stakeholders by the Lab. In other words, the ap-

The screenshot shows a web-based application window titled 'BizNetCompliance'. The main navigation bar includes 'Administration', 'BizNetCompliance', 'Business Network Management', 'Compliance Management', 'Compliance Requirements', 'Monitoring Capabilities', and 'Dashboard (metric)'. Below this is a toolbar with 'Search', 'New', 'Edit', 'Delete', and a 'Run Monitorability Check' button. The central area is titled 'Action List to Restore Monitorability' and contains a table with the following data:

Monitoring requirement	Required action
occurrence block: AdmitPatient_Treatment	no action required
occurrence block: DestroyBloodSample_BloodT	create monitoring capability
timestamp block: AdmitPatient_Treatment	no action required
timestamp block: DestroyBloodSample_BloodT	create monitoring capability

Below this table is another table with the following data:

ID	type	classification	Business Network	block A (operand)	block B (operand)	isMonitorable?
6	precedes	atomic	TextileIndustrialDistrict	A_outsourcingProc	Bout_outsourcingProc	No
7	precedes	atomic	TextileIndustrialDistrict	AA_processC	C_processC	No
8	precedes	atomic	TextileIndustrialDistrict	C_processC	BB_processC	No
9	segregatedFrom	atomic	Healthcare	AdmitPatient_Treatment	DischargePatient_Treatment	Yes
10	leadsTo	atomic	Healthcare	AdmitPatient_Treatment	DestroyBloodSample_Treatment	Yes

Fig. 6. List of actions to restore compliance monitorability in BizNetCompliance

plication of the mechanism of Algorithm 4 returns an empty set of actions, as all required monitoring capabilities to monitor the compliance requirements defined in the BN are now available.

5 Prototype Implementation

Our framework for alignment of process compliance and monitoring requirements is implemented in the prototype **BizNetCompliance**¹. It allows the business process expert to specify the structure of BNs, in terms of actors, business processes, and related blocks. The compliance expert can specify the compliance and monitoring requirements in the BN and run the alignment mechanisms in case of BN evolution.

The core of the application is constituted by two components to manage business networks structure and compliance requirements. The tool can be extended by plug-ins. A plug-in we developed allows the evaluation of compliance alignment metrics, which we briefly describe later in this section.

As far as network structure is concerned, the tool allows business process experts to specify the aspects of business processes relevant for compliance, i.e. the block structure, excluding control flow and connectors; alternatively, the tool can extract the block structure from processes specified in BPMN. The extensions to BPMN required by our framework, i.e. to specify outsourced blocks and placeholders, exploit the extension mechanism of the *Task* element proposed in BPMN 2.0.

BizNetCompliance is a Web-based information system that can be accessed using a browser. It has been partly realized using a state of the art model-driven tool

¹ <https://sites.google.com/site/biznetcompliancemonitor/>

for enterprise systems application development and integration². Fig. 6 shows a snapshot of the output of the tool after the execution of the mechanism in Alg. 3 to preserve alignment when the compliance requirement `AdmitPatient` leads to `DestroyBloodSample` is introduced in the BN of our running example. The output in this case is constituted by a set of monitoring capabilities currently missing in the BN, which should be implemented to preserve the alignment between the new compliance requirement and its related monitoring requirements (see Fig. 4).

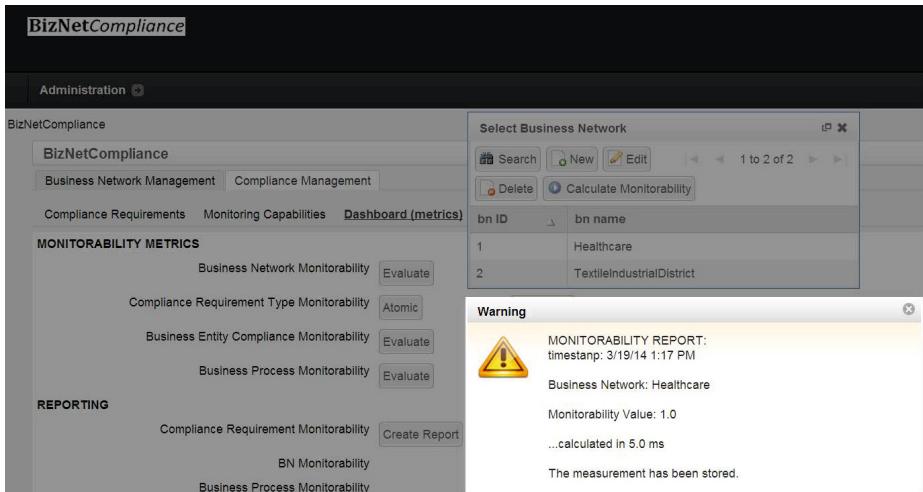


Fig. 7. Dashboard for compliance monitorability and example report in BizNetCompliance

As anticipated, we extended the core of the tool with a plug-in to evaluate compliance-monitoring alignment metrics. Such metrics give a quantitative evaluation of the degree of alignment between compliance and monitoring requirements and are important for a variety of reasons. In the design phase, and in respect of network evolution, alignment metrics are important *ex-ante*, i.e. to assess the impact on the BN of a particular evolution, and *ex-post*, i.e. to assess the status of the BN from the point of view monitorability after evolution has occurred. In the ex-ante perspective, alignment metrics can help evaluating, for instance, the impact of outsourcing a given process. While outsourcing may be beneficial from an economic point of view, it may also disrupt the compliance monitorability of the BN in case blocks involved by many compliance requirements are outsourced to a business entity with limited monitoring capabilities. In the ex-post perspective, alignment metrics give a synthetic view of the compliance level of a BN. Higher values of such metrics are likely to increase the appeal of the BN to potential customers or potential business partners.

² www.mendix.com

The basic metric we consider is the monitorability of an individual compliance requirement $MON(cr)$. Given the number D of monitoring capabilities induced (desired) by a compliance requirement and the number A of such capabilities currently available within the BN where the compliance requirements is required, we have $MON(cr) = A/D$, with $0 \leq MON(cr) \leq 1, \forall cr$.

Given the basic metric, we defined metrics at the level of business network, processes, entities, and type of compliance requirement:

- Business network alignment: average monitorability of all compliance requirements defined in a BN;
- Business process alignment: average monitorability of all compliance requirements involving a particular process, i.e. using at least one of the blocks of the process as operand;
- Actor compliance alignment: average monitorability of all compliance requirements involving all processes contributed by a given business entity (actor);
- Compliance requirement type alignment: average monitorability of all compliance requirements of a given type.

The metrics do not aim at being exhaustive, but they rather aim at giving a glimpse of the potential of our framework in giving a quantitative perspective about the compliance-monitoring alignment in a BN.

Figure 7 shows a snapshot of the dashboard to monitor the compliance monitorability metrics of a BN and an example evaluation. The system also enables users to print detailed reports of the monitorability gaps in the BN, e.g. list of missing monitoring capabilities to achieve full monitorability of existing compliance requirements.

We evaluated the performance of our tool in respect of the time required to (i) execute the alignment algorithms and (ii) calculate the values of metrics defined above. The execution of the outsourcing alignment mechanism of Alg. 4 is the most critical from a performance point of view, as its execution time grows polynomially with the number of compliance requirements predicated on the outsourced block(s) and the depth level of the outsourcing chain. We created a test-bed involving example processes with up to 20 outsourced activities and up to 5-level deep outsourcing chains, with complex compliance requirements obtained as combination of the atomic patterns of Table 1 predicated on up to 20 blocks (activities). On a standard desktop machine equipped with a quad-core CPU our tool returns the list of monitoring capabilities to be created to maintain alignment in less than 50ms. Note that the runtime performance of our tool is not particularly critical as the tool is intended to be used at design time.

6 Related Work

Compliance management of internal business processes has received a large deal of attention recently in the areas of compliance requirements specification [23], monitoring [10], and diagnostics [8,22]. A review of compliance management frameworks can be found in [7]. A review of emerging research challenges in

compliance management is presented in [1]. Among others, the authors identify (i) difficulties in creating evidence of compliance, (ii) frequent changes in regulations, and (iii) lack of IT support as major challenges for companies implementing compliance. We address the above challenges in the context of cross-organizational compliance management by (i) providing a framework to assess the monitoring capabilities required to guarantee compliance when (ii) new compliance requirements are deployed. We show the feasibility of our approach by (iii) discussing a prototype implementation.

Concerning compliance management in collaborative scenarios, Reichert et al. [16] identify the challenges of compliance management in cross-organizational processes. A subset of such challenges concerns the dynamic aspect of BNs and, in particular, the need for a compliance management framework to adapt to changes of processes and requirements within BNs. This paper contributes towards closing this gap, by considering the alignment between compliance and monitoring requirements for specific types of change, i.e. out/backsourcing and deployment of new compliance requirements.

Our literature review identified three main contributions specific to the case of compliance management in collaborative scenarios [15,20,24].

The issue of cross-organizational compliance is tackled using interaction models in [15]. The authors propose a method to guarantee global compliance requirements on a collaboration by looking at the interaction models derived by matching the process public views of collaborating parties. Compliance is satisfied if the requirements fit the generated interaction models. The approach deals with static collaborations, as it does not explicitly consider either requirements or collaboration structural changes.

The work presented in [20] defines a model for the event-based compliance checking of contract-based collaborations. Although the work focuses on business conversations, i.e. the messages exchanged during a collaboration, rather than business processes, it proposes concepts that we have adopted in our framework. Similarly to the monitoring capabilities of our framework, each business operation used in a conversation should generate a set of monitoring events to allow the compliance checker to run compliance control. Moreover, events are timestamped, to check those properties referring to the order of messages and operations.

A declarative way to model cross-organizational processes is based on the notion of *commitments* of the agents involved in the collaboration [24,6]. Commitments may be used to specify and verify compliance requirements [24]. This approach prevents compliance to predicate on the structural aspects of business networks and fits the case in which formalized business process models are not available to describe the collaboration.

As far as compliance metrics are concerned, the paper [17] defines the metric of design-time compliance distance, where the structure of a business process is assessed against the ideal situation represented by the satisfaction of compliance requirements. In our framework, compliance is ideal when all the monitoring capabilities required by a compliance requirement are available within the business network.

7 Conclusions and Outlook

In this paper we presented a framework for aligning compliance and monitoring requirements in evolving business networks. The framework comprises a conceptual model of business networks, compliance and monitoring requirements, and their evolution and mechanisms to ensure alignment of compliance and monitoring requirements. We discussed a *concretization* of the framework for specific implementation choices regarding the modelling of compliance requirements and business process outsourcing. Such implementation choices have driven the prototype implementation.

A first extension of this work will concern the evaluation of our framework with practitioners in a real world BN. In particular, we aim at evaluating (i) the extent to which our framework is able to cover all relevant compliance and monitoring requirements in the chosen scenario and (ii) the reduced effort for the compliance expert in managing the complexity of evolving requirements. Such an evaluation will involve working sessions and possibly controlled experiments with practitioners actively engaging with our prototype implementation in modelling compliance requirements and assessing the impact of evolution on the monitorability of a BN. Specifically, we will consider the case of a BN of mobility services that we have already engaged for testing tools for formulating and communicating network-based service strategy [18].

The models proposed in this paper can be extended, by looking at new types of network or compliance evolution, and by considering the run-time view of our framework, e.g. monitoring of executing processes, sampling of relevant compliance data, root-cause analysis of compliance requirements violations. The alignment metrics allow the definition of enhanced BN management features, such as dynamic partner selection or process delegation based on the maximization of the alignment between process compliance and monitoring requirements.

References

1. Abdullah, N.S., Sadiq, S., Indulska, M.: Emerging challenges in information systems research for regulatory compliance management. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 251–265. Springer, Heidelberg (2010)
2. Ardagna, D., Baresi, L., Comai, S., Comuzzi, M., Pernici, B.: A service-based framework for flexible business processes. *IEEE Software* 28(2), 61–67 (2011)
3. Beheshti, S.-M.-R., Benatallah, B., Motahari-Nezhad, H.R.: Enabling the analysis of cross-cutting aspects in ad-hoc processes. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 51–67. Springer, Heidelberg (2013)
4. Camarinha-Matos, L., Afsarmanesh, H.: A comprehensive modeling framework for collaborative networked organizations. *Journal of Intelligent Manufacturing* 18(5), 529–542 (2007)
5. Comuzzi, M., Vonk, J., Grefen, P.: Measures and mechanisms for process monitoring in evolving business networks. *Data Knowl. Eng.* 71, 1–28 (2012)
6. Desi, N., Chopra, A.K., Singh, M.P.: Amoeba: A methodology for modeling and evolving cross-organizational business processes. *ACM TOSEM* 29(2) (2009)
7. El Kharbili, M.: Business process regulatory compliance management solution frameworks: A comparative evaluation. In: APCCM, pp. 23–32 (2012)

8. Elgammal, A., Turetken, O., van den Heuvel, W.-J., Papazoglou, M.: Root-cause analysis of design-time compliance violations on the basis of property patterns. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 17–31. Springer, Heidelberg (2010)
9. Fdhila, W., Rinderle-Ma, S., Reichert, M.: Change propagation in collaborative processes scenarios. In: Proc. IEEE CollaborateCom, pp. 452–461 (2012)
10. Giblin, C., Müller, S., Pfitzmann, B.: From regulatory policies to event monitoring rules: Towards model-driven compliance automation. Technical Report 99682, IBM Research GmbH, Zürich (2006)
11. Grefen, P., Eshuis, R., Mehandijev, N., Kouvas, G., Weichert, G.: Internet-based support for process-oriented instant virtual enterprises. In: IEEE Internet Comput., pp. 30–38 (2009)
12. Grefen, P., Ludwig, H., Angelov, S.: A three-level framework for process and data management of complex E-Services. IJCIS 12(4), 487–531 (2003)
13. Grefen, P., Ludwig, H., Dan, A., Angelov, S.: An analysis of web services support for dynamic business process outsourcing. Information & Software Technology 48(11), 1115–1134 (2006)
14. Karagiannis, D., Mylopoulos, J., Schwab, M.: Business process-based regulation compliance: The case of the sarbanes-oxley act. In: IEEE Int. Requirements Engineering Conference, pp. 315–321 (2007)
15. Knuplesch, D., Reichert, M., Fdhila, W., Rinderle-Ma, S.: On enabling compliance of cross-organizational business processes. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 146–154. Springer, Heidelberg (2013)
16. Knuplesch, D., Reichert, M., Mangler, J., Rinderle-Ma, S., Fdhila, W.: Towards compliance of cross-organizational processes and their changes. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBP, vol. 132, pp. 649–661. Springer, Heidelberg (2013)
17. Lu, R., Sadiq, S., Governatori, G.: Measurement of compliance distance in business processes. Information Systems Management 25, 344–355 (2008)
18. Lüftenerger, E., Comuzzi, M., Grefen, P.: The service-dominant ecosystem: Mapping a service dominant strategy to a product-service ecosystem. In: Camarinha-Matos, L.M., Scherer, R.J. (eds.) PRO-VE 2013. IFIP AICT, vol. 408, pp. 22–30. Springer, Heidelberg (2013)
19. Mendling, J., Reijers, H., van der Aalst, W.: Seven process modeling guidelines (7PMG). Information and Software Technology 52(2) (2010)
20. Molina-Jimenez, C., Shrivastava, S., Strano, M.: A model for checking contractual compliance of business interactions. IEEE Transactions on Services Computing 5(2), 276–289 (2012)
21. Montali, M., Maggi, F., Chesani, F., Mello, P., van der Aalst, W.: Monitoring business constraints with the event calculus. ACM Transactions on Intelligent Systems and Technology 5 (2013)
22. 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)
23. Sadiq, W., 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)
24. Telang, P., Singh, M.: Specifying and verifying cross-organizational business models: An agent-oriented approach. IEEE Transactions on Services Computing 5(3), 305–318 (2012)

TAGER: Transition-Labeled Graph Edit Distance Similarity Measure on Process Models

Zixuan Wang, Lijie Wen, Jianmin Wang, and Shuhao Wang

School of Software, Tsinghua University, Beijing 100084, P.R. China
iamwangzixuan@hotmail.com, wenlj00@mails.tsinghua.edu.cn,
jimwang@tsinghua.edu.cn, shudiwsh2009@gmail.com

Abstract. Although several approaches have been proposed to compute the similarity between process models, they have various limitations. We propose an approach named TAGER (Transition-lAbeled Graph Edit distance similarity MeasuRe) to compute the similarity based on the edit distance between coverability graphs. As the coverability graph represents the behavior of a Petri net well, TAGER, based on it, has a high precise computation. Besides, the T-labeled graphs (an isomorphic graph of the coverability graph) of models are independent, so TAGER can be used as the index for searching process models in a repository. We evaluate TAGER from efficiency and quality perspectives. The results show that TAGER meets all the seven criteria and the distance metric requirement that a good similarity algorithm should have. TAGER also balances the efficiency and precision well.

Keywords: Business Process Model, Transition-labeled Graph, Edit Distance, Behavioral Similarity.

1 Introduction

Repositories with hundreds and even thousands of process models become increasingly common [8] and the organizations reach higher levels of Business Process Management (BPM) maturity [9]. There is an urgent requirement for searching the specific models among the process repository. Although a variety of techniques that can manage these repositories have been proposed already, the requirement for precise similarity measure is becoming more and more urgent. There are several approaches to measure the similarity between process models. A general classification for those algorithms [12] is as follows, according to which kind of information these algorithms are mainly based on.

- Text similarity: based on a comparison of the labels that appear in process models (e.g., task labels, event labels), using either syntactic or semantic similarity metrics, or a combination of them.
- Structural similarity: based on the topology of process models, possibly taking text similarity into account as well.
- Behavioral similarity: based on the execution semantics of process models.

Because the similarity measures based on text or structure only consider the label set or topological structure, they lack lots of information. Comparatively, the similarity measures based on behavioral features can have a better performance and provide a more convincing result. The existing behavioral similarity algorithms have various advantages. However, they have limitations, too. The PTS similarity measure algorithm [11], proposed by Jianmin Wang in 2010, is based on labeled Petri net. It defines three types of principal transition sequences and computes a result by weighting the similarity value of each type. But this approach considers the loop structure and other structures separately. As a result, it breaks the behavior semantics and it cannot handle Petri nets with loop structures effectively. TAR similarity algorithm [15] represents a model's behavior by transition adjacency relations. It computes the similarity by Jaccard coefficient of two TAR sets but cannot tackle the long-distance dependences between transitions. BP algorithm [14] gives the similarity based on behavioral profiles between transitions. However, it cannot distinguish the loop and parallel structures. Furthermore, it cannot deal with process models with invisible tasks.

TAGER, as a behavioral similarity measure, chooses Petri net as its modeling notation. The Petri net analysis tool, coverability graph [10], can represent the behavioral features of a Petri net well. However, the transition labels as the most useful information are stored on a coverability graph's edges. So we define the Transition-labeled graph (T-labeled graph for short), the isomorphic graph of a coverability graph, to emphasize the transition label on the edges of a coverability graph. Because graph edit distance is a classical method to measure the difference between two graphs, so TAGER choose to measures the similarity between T-labeled graphs by using their graph edit distance. And TAGER modifies and extends the traditional edit operations to get a precise similarity. The graph edit distance generally has been used in structural similarity measure, however, we use it based on the coverability graph which is the behavioral feature of the Petri net. Besides, compared to the traditional edit operations, we redefine substitute operation and consider some common behavior patterns. Our main contributions in this paper are as follows:

1. We propose an efficient and effective approach named TAGER to compute the behavioral similarity of process models. It is based on the edit distance of T-labeled graphs, which contain all behavior information of Petri nets.
2. To improve the precision, we redefine the graph edit operations [5] according to process models' structural features. We take the type and scale of vertices into account when redesigning the vertex substitution operation.
3. We define the similarity measure by combining the model scale with graph edit distance and give a sound computation approach with the best weight arrangement obtained by experiments.

The remainder of the paper is structured as follows. Section 2 formulates the problem and introduces the relevant concepts which will be used throughout the paper. Section 3 presents our algorithm which is used to measure the similarity. Section 4 presents the experimental evaluation of our algorithm. Section 5 concludes this paper and sketches some future work.

2 Preliminaries

This section presents the notions of process models used throughout this paper.

2.1 Petri Net

Petri net was originally introduced by Carl Adam Petri [7] as a tool for modeling basic structures like sequence, conflict, parallel and loop which can be used to study the parallel, distributed and random system [6]. A Petri net is consisted of places and transitions. For each node, a unique identifier (like P_1 or T_2 in Figure 1) is associated. For the remainder of this paper, we assume that the set of possible identifiers is denoted as U .

Definition 1. (Petri net) Let $P, T \subseteq U$ be finite and disjoint sets such that $PUT \neq \emptyset$ and $P \cap T = \emptyset$, let $F \subseteq (P \times T) \cup (T \times P)$. The tuple $N = (P, T, F, M_0)$ is a system, where P is the set of places, T is the set of transitions, F is the set of arcs and M_0 as a mapping from P to \mathbb{N} is the initial marking of the net. \mathbb{N} is the set of non-negative integers, i.e., $\{0, 1, 2, \dots\}$.

According to Definition 1, we introduce labeled Petri net. For each transition, a unique label is associated which indicates the action name.

Definition 2. (Labeled Petri net) Let $N = (P, T, F, M_0)$ be a Petri net, the tuple $N = (P, T, F, M_0, A, l)$ is a labeled Petri net, where A is a finite set of action names, function l is a mapping from T to $A \cup \{\varepsilon\}$, ε represents an empty label of a dumb or silent transition.

Figure 1 gives two samples of labeled Petri nets, i.e., M_0 and M_4 . For example, P_0, P_1, P_2 and P_3 are the places of M_4 and T_0, T_1, T_2 and T_3 are the transitions of M_4 .

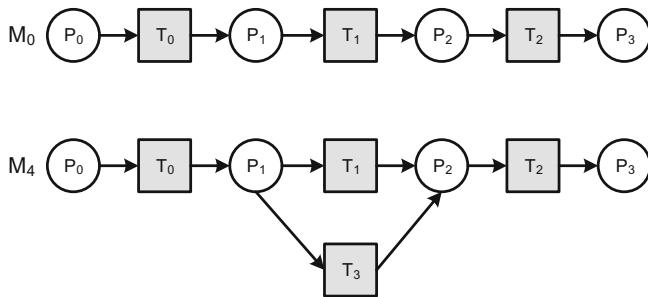


Fig. 1. Two sample labeled Petri nets

Definition 3. (Workflow net) A Petri net $N = (P, T, F, M_0)$ that models the control-flow dimension of a workflow is called a workflow net (WF-net for short), which should satisfy the following conditions:

- There is one unique source place i such that $\bullet i = \emptyset$;
- There is one unique sink place o such that $o\bullet = \emptyset$;
- Every node $x \in P \cup T$ is on a path from i to o .

2.2 Coverability Graph

Coverability graph is an effective model to represent a Petri net's behavioral features. The markings of the Petri net are mapping to the vertices of the coverability graph and the transitions of the Petri net are mapped to the edges of the coverability graph. A coverability graph differes from the topological structure of a Petri net by having the behavioral features. We can compute a coverability graph by the following steps [10].

Definition 4. (*Coverability graph*) Let $N = (P, T, F, M_0, A, l)$ be a labeled Petri net, let $S = (N, I, O)$ be a net system, I is the system's initial state, whereas the marking O is its successful terminal state. Let $V \subseteq \mathbb{M}$ where \mathbb{M} is the marking set of N , let $E \subseteq (V \times T \times V)$ be a set of Transition-labeled arcs, and let $G = (V, E)$ be a graph which can be constructed as follows:

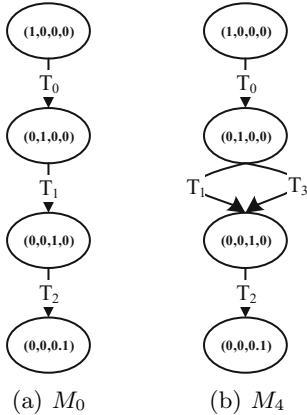
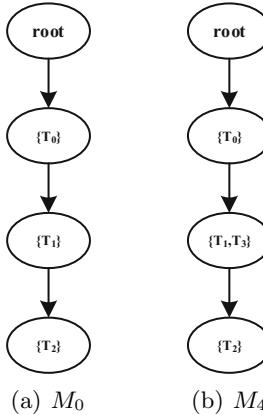
1. Initially, $V = \{I\}$ and $E = \emptyset$.
2. Take an extended marking (the new marking after firing any enabled transition at current marking) $M \in V$ and a $t \in T$ such that $M[t]$ and no extended marking M_1 exists with $(M, t, M_1) \in E$. Let $M_2 = M_1 - \bullet t + t\bullet$. Add M_3 to V and (M, t, M_3) to E , where for every $p \in P$:
 - (a) $M_3(p) = \omega$, if there is a node $M_1 \in V$ such that $M_1 \leq M_2$, $M_1(p) < M_2(p)$, and there is a directed path from M_1 to M in G ;
 - (b) $M_3(p) = M_2(p)$, otherwise.
- Repeat this step until no new arcs can be added.

Graph G is called a coverability graph of a net system S .

For example, we can capture the coverability graphs of M_0 and M_4 in Figure 1, which are shown in Figure 2.

3 The New Algorithm TAGER

Our algorithm named TAGER is based on coverability graph which can analyze a Petri net without losing structural information. As the graph edit distance is a classical approach to measure the difference between two graphs, we choose it as the crucial factor on measuring two process models' similarity. In this section, we give an overview of TAGER and introduce the details of the crucial computing steps. We analyze the time complexity of TAGER and discuss its limitations in the end of this section.

**Fig. 2.** Corresponding coverability graphs**Fig. 3.** Corresponding T-labeled graphs

3.1 The Algorithm Overview

To compute the graph similarity of two process models, we need to find a mapping that gives a maximal similarity match and a minimal graph edit distance. The TAGER algorithm computes the similarity between two process models expressed as Petri nets in the following steps:

1. Building the coverability graph for each Petri net;
2. Converting the coverability graphs to T-labeled graphs, which will be defined in the following subsection.
3. Using the modified greedy algorithm to compute the graph edit distance between the two T-labeled graphs.
4. Computing the similarity with the weights of the edit distance and the scale of graphs.

We can finish step 1 and step 2 according to Definition 4 in Section 2 and Definition 5 in Section 3.2 respectively. Step 3 can be computed on the foundation we give in Section 3.3 and via the method we will introduce in Section 3.4. Step 4 will be shown in Section 3.5.

3.2 The T-Labeled Graph

As TAGER is defined as a behavioral measure, we need the transition information which are stored on the edges in the coverability graph. Using the coverability graph directly will make the computation of the edit distance complex, so we transform the coverability graph to its isomorphic graph, which we named *T-labeled graph*. We just move the labels of transitions to their successive vertices.

Definition 5. (*T-labeled graph*) Let $N = (P, T, F, M_0, A, l)$ be a labeled Petri net and $G_{\text{coverability}} = (V, E)$ be its coverability graph, where V stores the current markings and E stores the transitions' labels. Let \mathcal{L} be the set of transition labels. We define *T-labeled graph* $G = (V, E, \lambda)$ as follows:

- V is the vertex set of $G_{coverability}$;
- E is the edge set of $G_{coverability}$;
- $\lambda : V \rightarrow 2^{\mathcal{L}}$ is a function that maps vertices to label sets.

Now all transitions' labels are stored in vertices and the edges are just be used as the connectors of vertices. Let $G = (V, E)$ be a graph and $n \in V$ be a node: $\bullet n = \{m | (m, n) \in E\}$ is the pre-set of n , while $n\bullet = \{m | (n, m) \in E\}$ is the post-set of n . Source node is a node with an empty pre-set and sink node is a node with an empty post-set. According to Definition 3, a T-labeled graph can only has one source node and one sink node. We can convert coverability graphs in Figure 2 to their T-labeled graphs, which are shown in Figure 3.

3.3 The Graph Edit Operations

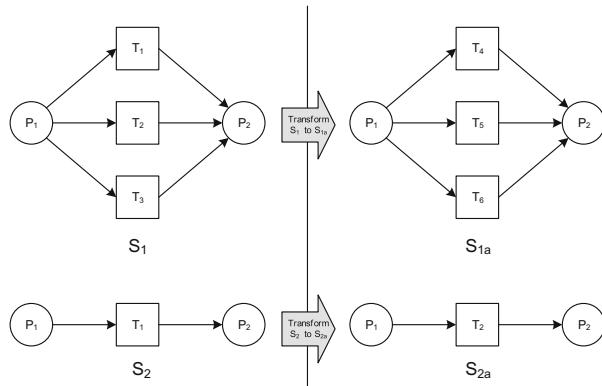
The graph edit distance is computed on the foundation of edit operations, in which we redefine in TAGER algorithm. Besides, we give the equation for computing the edit distance between two graphs.

The Edit Operations. The edit operations which have been mentioned in this paper include vertex insertion/deletion, vertex substitution and edge insertion/deletion. The cost of vertex insertion/deletion is assigned one in TAGER. Furthermore, the traditional computation of the cost in the other two edit operations ignores the influence of label set and the weight of edges. Therefore, we partly redefine these two edit operations as follows.

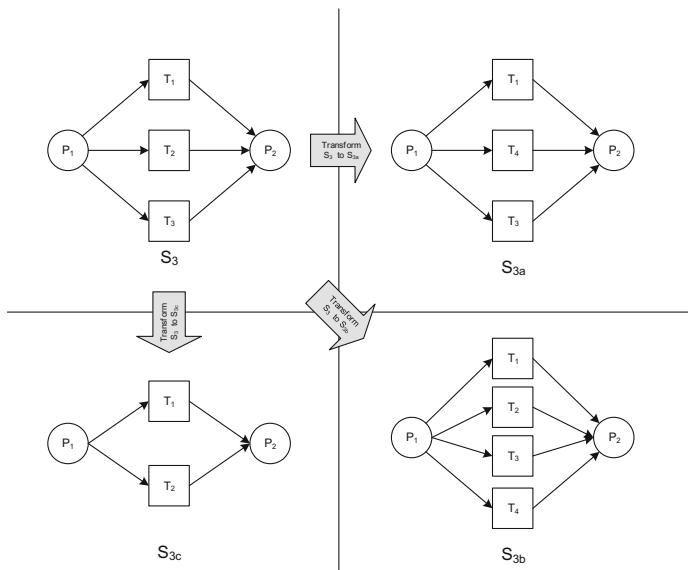
Vertex Substitution. Generally, the cost of a vertex substitution from v_1 to v_2 is equal to the string edit distance between the two vertices' labels. However, in TAGER, a vertex's label in a T-labeled graph may be consisted of several transitions' labels which is a label set. We take the label set into account when we compute the cost of vertex substitution.

$$\begin{aligned} \text{cost}(v_1, v_2)^{\text{sub}} &= \frac{k \cdot \text{abs}(|v_1.l| - |v_2.l|)}{|v_1.l| + |v_2.l|} \\ &\quad + \frac{|v_1.l \cup v_2.l| - |v_1.l \cap v_2.l|}{|v_1.l \cup v_2.l|} \end{aligned} \tag{1}$$

$v_1.l$ and $v_2.l$ are the label sets of vertex v_1 and v_2 respectively, and $\text{abs}(x)$ represents the absolute value of x . k is a user-defined factor used to weight the effect by scale difference, as the experimental best value we set k to 2. The first part of Equation (1) represents the scale difference of two vertices. As conflict/parallel structure will bring labels of multiple transitions into one vertex, the label set of a vertex may contain more than one label. The second part is the pure substitution cost. This part of cost is influenced by two vertices' label sets, which is denoted as the difference between the two label sets. We exemplify the soundness of the substitution operation with two examples on Petri net for directly perceived feeling. Readers can compute the T-labeled graphs according to Definition 5 instead.

**Fig. 4.** Simple case of vertex substitution

Example 1. Figure 4 shows that if the scales of the vertices are equal, the upper bound of the substitution cost will be one. Converting S_1 to S_{1a} and S_2 to S_{2a} have the same cost.

**Fig. 5.** Complex case of vertex substitution

Example 2. In Figure 5, we present a complex situation when the scales of the vertices are different. Converting S_3 to S_{3a} , S_3 to S_{3b} and S_3 to S_{3c} requires a substitution, an insertion and a deletion of a vertex respectively in the original

Petri net. However, only one vertex substitution occurs in the corresponding T-labeled graphs, from vertex label set $\{T_1, T_2, T_3\}$ to $\{T_1, T_4, T_3\}$, $\{T_1, T_2, T_3, T_4\}$ and $\{T_1, T_2\}$ respectively. The costs of the three substitutions will be $0+2/4+0 = 1/2$, $2/7 + 1/4 + 0 = 15/28$ and $2/5 + 1/3 + 0 = 11/15$ respectively.

Compared to Figure 4, we can see that a partial substitution costs less than a whole one. Besides, if the reference model is unchanged, a deletion affects their similarity bigger than an insertion. This shows that our vertex substitution can distinguish all general situations.

Edge Insertion/Deletion. In most situations, the cost of edge insertion/deletion is assigned one. In our approach, we assign weight to an edge according to its type and structure in a T-labeled graph. Firstly, we determine if an edge e is a *loopedge* (edges related to the loop structure) or a *skipedge* (edges related to the conflict structure). If so, the weight of e is set as the max span of the sequential part from source place to sink place of the T-labeled graph where e is repeated or skipped over, denoted as $span_{loop}(e)$ or $span_{conflict}(e)$. Otherwise, it is set as one. Therefore, the cost of inserting or deleting an edge e is defined as:

$$cost(e)^{ins/del} = \max(span_{loop}(e), span_{conflict}(e), 1) \quad (2)$$

For the given model in Figure 3, the third vertex pair ($\{T_1\}$, $\{T_1, T_3\}$) requires a substitution operation. The cost of scale difference part is $2 \times (2-1)/(2+1) = 0.667$, while the cost of node label set difference part is $(2-1)/2 = 0.5$. As we take into account the conflict span only when its scale is more than one or it is a general edge, the cost of conflict or loop part is 0. Therefore, their substitution cost is $0.667 + 0.5 = 1.167$.

3.4 The Graph Edit Distance

One graph can be converted to another using the edit operations described above. If we assign appropriate weight to the cost of each operation, we can get the total cost of converting one graph to another, which is so-called graph edit distance.

Definition 6. (Graph edit distance) Let $G_1 = (N_1, E_1, \lambda_1)$ and $G_2 = (N_2, E_2, \lambda_2)$ be two T-labeled graphs. Let $M : N_1 \nrightarrow N_2$ ¹ be a function that maps nodes in G_1 to nodes in G_2 . Let $dom(M) = \{n_1 \in N_1 | (n_1, n_2) \in M\}$ be the domain of M and $cod(M) = \{n_2 \in N_2 | (n_1, n_2) \in M\}$ be the codomain of M . Given $n \in N_1 \cup N_2$, n is substituted iff $n \in dom(M)$ or $n \in cod(M)$. SubNodes is the set of all substituted nodes. A node $n_1 \in N_1$ that is deleted from G_1 or inserted in G_1 iff it is not substituted. SkipNodes is the set of all inserted and deleted nodes in G_1 . SkipEdges is the set of all inserted and deleted edges in G_1 . Let w_{subn} , w_{skipn} and w_{skipe} be the weights that we assign to substituted nodes, inserted or deleted nodes, inserted or deleted edges respectively.

¹ We use \nrightarrow to represent the partial injective mapping.

The graph edit distance induced by the mapping M is defined as:

$$\begin{aligned} \text{editDistance}(G_1, G_2) = & wskipn \cdot |\text{SkipNodes}| + wskipe \cdot |\text{SkipEdges}| \\ & + wsubn \cdot |\text{SubNodes}| \end{aligned}$$

$\text{editDistance}(G_1, G_2)$ represents the graph edit distance between two T-labeled graphs G_1 and G_2 . As the experimental best performance, we take $k = 2.0$, $wskipn = 0.8$, $wsubn = 1.8$, $wskipe = 0.5$. And all these parameters can be defined by users. For example, we compute the graph edit distance of the two graphs in Figure 3. To convert T-labeled graph M_0 to M_4 , we need to substitute $\{T_1\} \rightarrow \{T_1, T_3\}$. The minimal graph edit distance between M_0 and M_4 in Figure 3 is induced by the following mapping:

$$\text{Mapping} : \{(\{T_0\}, \{T_0\}), (\{T_1\}, \{T_1, T_3\}), (\{T_2\}, \{T_2\})\}$$

The graph edit distance between M_0 and M_4 is $(2/3+1/2)*1.8 = 2.1$. We use the Greedy algorithm [2] to guarantee that the graph edit distance between the two graphs is the minimal possible distance.

3.5 The Similarity Computation

Both graph edit distance and the scale of graphs should be considered when we compute the similarity. Thus, we define the graph similarity as follows.

Definition 7. (Graph similarity) Let $G = (V, E, \lambda)$ represents the T-labeled graph we get from the previous step. $|G|$ is the scale of T-labeled graphs G , which is the total sum of $|V|$ and $|E|$. The similarity measure of two T-labeled graphs G_1 and G_2 is defined as follows:

$$\text{Similarity}(G_1, G_2) = 1 - \frac{\text{editDistance}(G_1, G_2)}{|G_1| + |G_2|} \quad (3)$$

The similarity can be viewed as the complement of differentiation. We can compute the differentiation by the edit distance, so we just need to consider the influence by model scale or graph scale.

3.6 The Algorithm Analysis of TAGER

We provide an integrated algorithm (see Algorithm 1) in this part, which basically follows the steps as we mentioned in Section 3.1.

The greedy algorithm part in TAGER, which can be used to compute the graph edit distance, is in $O(n^3)$ time complexity where n is the number of nodes of the largest graph.

As we consider all the possible factors that will influence the measure so that we can provide a precision similarity measure value. The time complexity of TAGER is $O(|V_1|^3 + |V_2|^3)$, $|V|$ is the number of vertices in a T-labeled graph.

The T-labeled graph will unfold the parallel structure. Although they will converge in the end, there is possibility of state explosion. For the same reason, our algorithm takes a longer time when the computed model contains a parallel structure with many branches.

Algorithm 1. TAGER algorithm

Input Two labeled Petri nets $N_1 = (P_1, T_1, F_1, M_{01}, A, l_1)$, $N_2 = (P_2, T_2, F_2, M_{02}, A, l_2)$

Output The similarity measure of N_1 and N_2

```

 $CoverabilityG_1 \leftarrow CoverabilityGraphBuilder(N_1)$ 
 $CoverabilityG_2 \leftarrow CoverabilityGraphBuilder(N_2)$ 
Use DFS to go through the whole graph to store the transition labels on all edges to the successive vertices.
 $T\text{-labeled}G_1 \leftarrow convert(CoverabilityG_1)$ 
 $T\text{-labeled}G_2 \leftarrow convert(CoverabilityG_2)$ 
Use the Remco's greedy algorithm [2] to compute the edit distance between two graphs with the edit operations and costs redefined in this paper.
 $editdis \leftarrow editDistance(T\text{-labeled}G_1, T\text{-labeled}G_2)$ 
 $similarity \leftarrow 1 - \frac{editdis}{|T\text{-labeled}G_1| + |T\text{-labeled}G_2|}$ 
return  $similarity$ 

```

4 Experimental Evaluation of TAGER

We have implemented TAGER in BeehiveZ² which is a business process data management system developed by Tsinghua University. In this section, we present the experimental evaluation of TAGER in both performance and quality perspectives. We evaluate the performance by applying TAGER on real-life dataset and evaluate the quality by comparing TAGER with other algorithms.

4.1 Experimental Setting

Our experimental datasets come from two parts, the artificial process models and the TC, DG and SAP reference models. These collections has 578 models (expressed in Petri nets) in total. The artificial models are mainly used to check whether TAGER can satisfy the properties that a good business process similarity measure should meet. The models coming from TC, DG and SAP can check whether TAGER meets the distance metric requirement.

The basic features of these models are summarized in Table 1. The following paragraphs briefly introduce the sources of the datasets.

- *The DG Dataset.* We have collected 94 real-life process models from Dongfang Boiler Group Co., Ltd. Dongfang Boiler is a major supplier of thermal power equipment, nuclear power equipment, power plant auxiliaries, chemical vessels, environmental protection equipment and coal gasification equipment in China (with more than 33% market share).
- *The TC Dataset.* We have collected 89 real-life process models from Tangshan Railway Vehicle Co., Ltd. (TRC). TRC is the oldest enterprise in Chinas railway transportation equipments manufacturing industry, with its predecessor constructing China's first railway line from Tangshan to Xugezhuang

² <https://code.google.com/p/beehivez/>

in 1881 during the westernization movement at the end of the Qing Dynasty. Today the company has built the train that has achieved the highest speed of Chinese railways - 394.3km/h.

- *The SAP Dataset.* SAP dataset is consist of SAP reference models. We delete such models that cannot satisfy the soundness property of WF-net. Then the dataset scale is decreased from 604 to 389.

Table 1. The features of three real-life reference model sets

Dataset	Size	Average			Minimum			Maximum		
		#transitions	#places	#arcs	#transitions	#places	#arcs	#transitions	#places	#arcs
SAP	389	4.465	7.504	11.514	1	2	2	21	31	56
DG	94	8.553	8.883	17.777	1	3	3	34	33	70
TC	89	11.472	10.281	22.933	6	5	11	28	29	58

We repeatedly extract every three models from the dataset, calculate the similarities between each pair of them and test whether they meet the triangle inequality property. The following two equations [4] are used to measure the triangle inequality property.

Definition 8. (*Triangle Inequality Property*) For any three models in one dataset, e.g. N_0, N_1, N_2 , the distances between any two of them are $dis(N_0, N_1)$, $dis(N_0, N_2)$, $dis(N_1, N_2)$ respectively, the triangle inequality property requires these distances to satisfy that the sum of any two of them should be bigger than the third one.

$$dis(N_0, N_1) = 1 - \text{Similarity}(N_0, N_1) \quad (4)$$

$$dis(N_0, N_1) = \begin{cases} \infty & \text{Similarity}(N_0, N_1) = 0 \\ 1/\text{Similarity}(N_0, N_1) - 1 & \text{otherwise} \end{cases} \quad (5)$$

The range of TAGER’s similarity value is $[0, 1]$, and the distances of those two equations both decrease with the increasing similarity value. The range of Equation (4) is $[0, 1]$, and the range of Equation (5) is $[0, \infty)$. As if there is more than one ∞ among the three distances, the triangle property should be satisfied.

4.2 Performance Evaluation on Real-Life Models

The performance of TAGER on real-life datasets are shown in Table 2. As transitions, places and arcs of models in the TC dataset is far more than those models in the other two datasets, the average time spent on building coverability graph and computing similarity value on the TC dataset are the highest of all. We can conclude that time consuming on a dataset is strongly related to the complexity of models in the dataset.

Table 2. The performance of TAGER

-	average time1 (ms) ¹	average time2 (ms) ²	satisfaction Eq.(4) ³	satisfaction Eq.(5) ⁴
SAP	4	3	1.0	0.95
DG	8	21	1.0	0.86
TC	11	33	1.0	0.96

¹ the average time spent on building the T-labeled graph for each process model in dataset.

² the average time spent on computing the similarity between two T-labeled graphs.

³ the satisfaction rate of triangle inequality property using Equation (4).

⁴ the satisfaction rate of triangle inequality property using Equation (5).

Moreover, TAGER on the three datasets has a 100% satisfaction on the triangle inequality property using Equation (4) while it performs well enough on the satisfaction using Equation (5).

Besides, TAGER has the 0.115ms average calculating time on 10,000 artificial models which are generated by BeehiveZ.

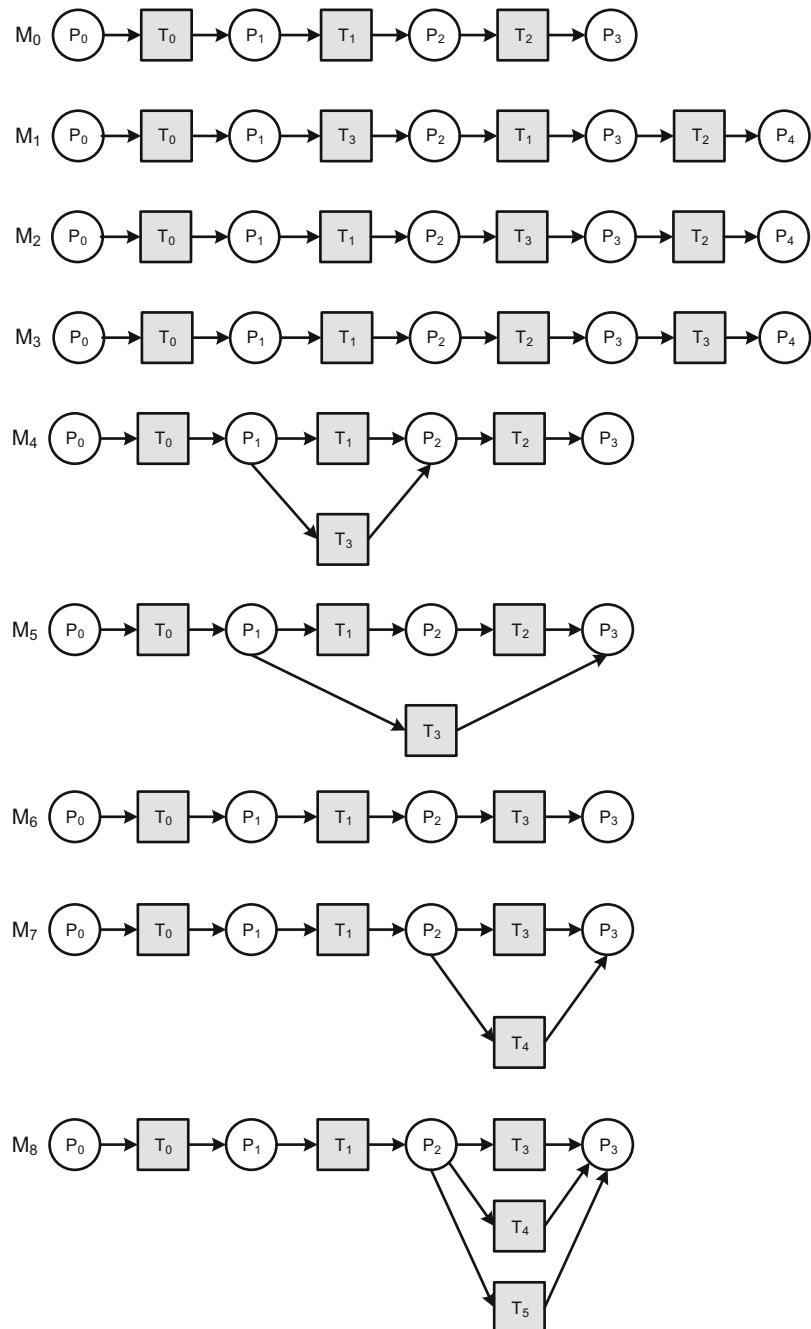
4.3 Quality Evaluation on Artificial Models

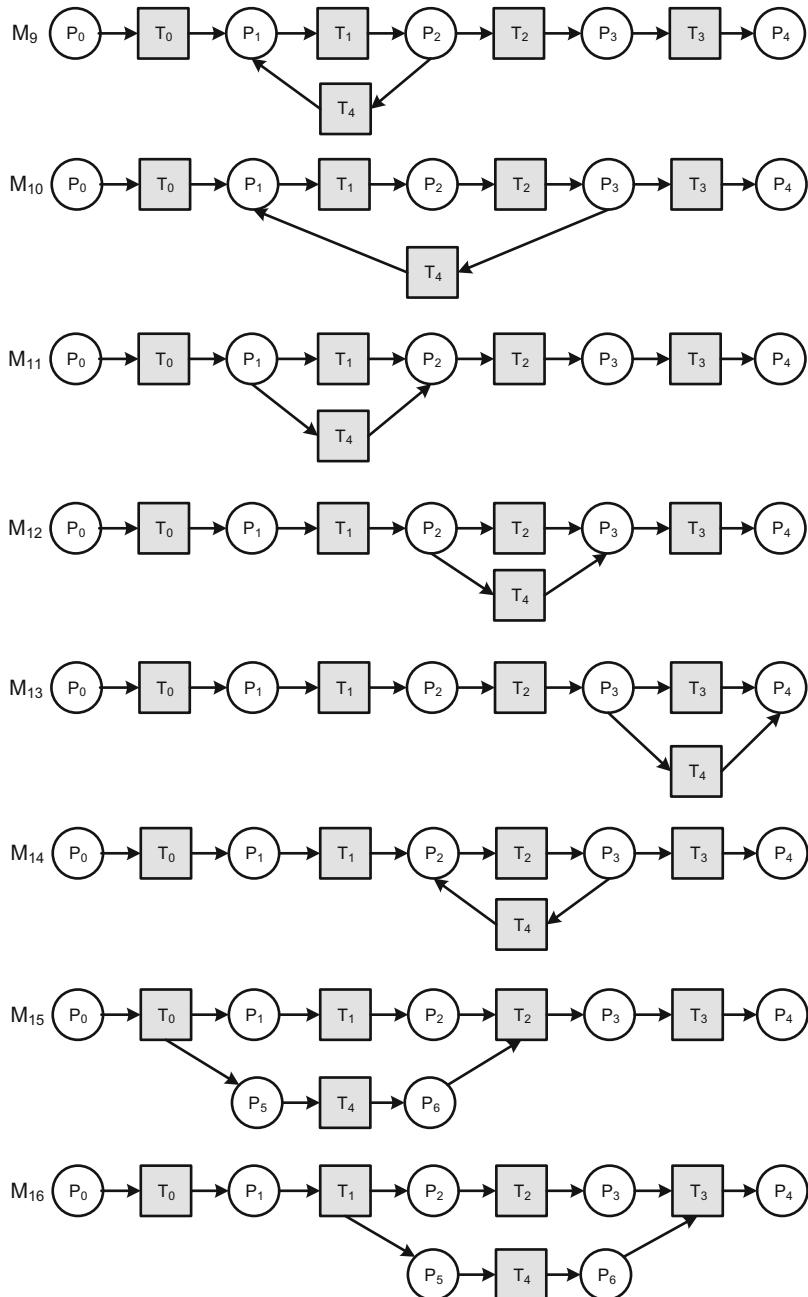
We evaluate TAGER by satisfying the properties that a good similarity algorithm should meet. The former six properties have been proposed in [13], we just use them directly. Besides, we propose a new property named parallel structure drift invariance. We provide the following models as shown in Figure 6 and Figure 7 to verify these properties. We compare TAGER with other algorithms such as TAR [15], PTS [11], SSDT [13], BP [14] and CF [3] which have a good performance on these properties, too. The artificial dataset (total 17 models) are collected from [1,13] and we have manually modified them slightly.

- **TAR:** Zha et al. [15] discuss a naive similarity measure, also called reference similarity, based on the set of transition adjacency relations.
- **PTS:** Wang et al. [11] propose a similarity measure based on principal transition sequences.
- **SSDT:** Wang et al. [13] define a shortest succession distance between tasks to compute the similarity value of two Petri nets.
- **BP:** Weidlich et al. [14] capture the behavior of a process model by examining dependencies between the execution of transitions.
- **CF:** Dijkman et al. [3] use causal footprints for capturing the precedence relations between transitions in a process model.

Property 1. Sequential structure drift invariance

If a Workflow net is sound, no matter where we add a sequential structure to the original sequential structure to derive new process models, they have the same similarity with the original one. The similarity should keep unchanged regardless the position of the added sequential structure.

**Fig. 6.** Artificial models from M_0 to M_8

**Fig. 7.** Artificial models from M_9 to M_{16}

For example, M_1 , M_2 , M_3 are derived from M_0 by adding transition T_3 . When these three models are compared with M_0 , they should have the same similarity value. The evaluation results on property 1 are shown in Table 3. We can see that except TAR, the other algorithms can satisfy this property.

Table 3. The evaluation results on property 1

sim	(M_0, M_1)	(M_0, M_2)	(M_0, M_3)
TAR	0.250	0.250	0.667
PTS	0.750	0.750	0.750
SSDT	0.813	0.813	0.813
BP	0.450	0.450	0.450
CF	0.286	0.286	0.286
TAGER	0.816	0.816	0.816

Table 4. The evaluation results on property 2

sim	(M_0, M_4)	(M_0, M_5)
TAR	0.500	0.667
PTS	0.889	0.778
SSDT	0.750	0.688
BP	0.460	0.525
CF	0.334	0.353
TAGER	0.873	0.800

Property 2. Negative correlation by conflict span

This property means that when modifying a model by adding a conflict structure on the sequential part, the longer the conflict span is, the greater the influence it makes on the similarity between the new model and the original one. The negative correlation reflects the special structure affecting the similarity.

For example, M_4 is more similar to M_0 than M_5 . The evaluation results on property 2 are shown in Table 4. We can see that except TAR, BP and CF, all others algorithms satisfy this property.

Property 3. Unrelated task regression

This property means the similarity between two models will decrease by adding more unrelated tasks.

For example, compare the similarity of M_0 and M_6 , M_7 , M_8 separately. M_6 replaces transition T_2 by T_3 , M_7 adds T_4 as a conflict structure at the same position and M_8 adds one more T_5 . The similarity between M_0 and M_6 , M_7 , M_8 should decrease. The specific result in Table 5 shows that except PTS, all other algorithms satisfy property 3. However, the results of TAR and BP are too low, the values of TAGER are close to the desirable ones. Besides, we prove that adding unlimited conflict structures will bring a limited influence on the similarity, and the lower bound is 0.672. It is a reasonable value for this situation.

Property 4. Negative correlation by loop length

This property means that adding a loop structure in the sequential structure, the similarity will decrease with the increasing scale of the loop structure.

For the models shown in Figure 6 and 7, the similarity between M_3 and M_9 is bigger than that between M_3 and M_{10} . The specific result in Table 6 shows that except TAR and PTS, all others algorithms satisfy this property.

Table 5. The evaluation results on property 3

<i>sim</i>	(M_0, M_6)	(M_0, M_7)	(M_0, M_8)
TAR	0.250	0.167	0.125
PTS	0.667	0.667	0.667
SSDT	0.750	0.680	0.611
BP	0.220	0.143	0.100
CF	0.392	0.340	0.285
TAGER	0.891	0.818	0.782

Table 6. The evaluation results on property 4

<i>sim</i>	(M_3, M_9)	(M_3, M_{10})
TAR	0.600	0.600
PTS	0.563	0.563
SSDT	0.760	0.720
BP	0.526	0.430
CF	0.259	0.242
TAGER	0.869	0.850

Property 5. Conflict structure drift invariance

This property means if adding a conflict structure in the sequential part, no matter where the conflict structure is added in the sequential structure, its influence to the similarity between the new model and the original model should keep unchanged.

For the models shown in Figure 6 and 7, the similarity between M_3 and M_{11} , M_3 and M_{12} , M_3 and M_{13} should be the same. The specific result in Table 7 shows that except TAR and CF, all others algorithms satisfy this property

Table 7. The evaluation results on property 5

<i>sim</i>	(M_3, M_{11})	(M_3, M_{12})	(M_3, M_{13})
TAR	0.600	0.600	0.750
PTS	0.917	0.917	0.917
SSDT	0.800	0.800	0.800
BP	0.514	0.514	0.514
CF	0.247	0.247	0.252
TAGER	0.900	0.900	0.900

Property 6. Loop structure drift invariance

This property means if adding a loop structure in the sequential part, no matter where the loop structure is added in the sequential structure, its influence to the similarity between the new model and the original model should keep unchanged.

For the models shown in Figure 6 and 7, the similarity between M_3 and M_9 , M_3 and M_{14} should be the same. The specific result in Table 8 shows that except PTS, all other algorithms satisfy this property.

Property 7. Parallel structure drift invariance

This is the new property we proposed in this paper. It means if adding a parallel structure in the sequential part, no matter where the parallel structure is added in the sequential structure, its influence to the similarity between the new model and the original model should keep unchanged.

Table 8. The evaluation results on property 6

<i>sim</i>	(M_3, M_9)	(M_3, M_{14})
TAR	0.600	0.600
PTS	0.563	0.625
SSDT	0.760	0.760
BP	0.526	0.526
CF	0.259	0.259
TAGER	0.869	0.869

Table 9. The evaluation results on property 7

<i>sim</i>	(M_3, M_{15})	(M_3, M_{16})
TAR	0.429	0.429
PTS	0.800	0.800
SSDT	0.760	0.760
BP	0.547	0.547
CF	0.226	0.226
TAGER	0.766	0.766

For the models shown in Figure 6 and 7, the similarity between M_3 and M_{15} , M_3 and M_{16} should be equal. The specific result in Table 9 shows that all algorithms satisfy this property.

Table 10 gives an overview of the quality evaluation on existing algorithms. It is obvious that TAGER and SSDT are the best ones. However, SSDT depends on another process model to compute the SSDT matrix, so it cannot be used as a kind of process model index.

Table 10. Summary of the quality evaluation

	Property1	Property2	Property3	Property4	Property5	Property6	Property7
TAR	✗	✗	✓	✗	✗	✓	✓
PTS	✓	✓	✗	✗	✓	✗	✓
SSDT	✓	✓	✓	✓	✓	✓	✓
BP	✓	✗	✓	✓	✓	✓	✓
CF	✓	✗	✓	✓	✗	✓	✓
TAGER	✓	✓	✓	✓	✓	✓	✓

5 Conclusion and Outlook

We provide a simple algorithm to calculate similarity between process models, which is based on the T-labeled graphs of Petri nets. As the graph edit distance is a classical approach to measure the difference between two graphs, we choose the edit distance as the crucial factor. The traditional operations work effectively in the situation containing no loop/conflict structures. We extend the substitution operation by considering the label sets of vertices and the cost of the loop/conflict structure. Experimental results show that TAGER has a good performance and meets all the properties while most other main-stream algorithms cannot. As parameters play an important role in TAGER and they are required to change frequently when datasets vary, we will focus on how to make the adjustment of parameters more automatically. Moreover, designing a new process model index based on TAGER is also part of our future work.

Acknowledgement. The research is supported by the MOE-CMCC research foundation project No. MCM20123011 and the special fund for innovation of Shandong, China No. 2013CXC30001.

References

1. Becker, M., Laue, R.: A comparative survey of business process similarity measures. *Computers in Industry* 63(2), 148–167 (2012)
2. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
3. Dijkman, R., Dumas, M., Dongen, B.V., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. *Information Systems* 36(2), 498–516 (2011)
4. Kunze, M., Weske, M.: Metric trees for efficient similarity search in large process model repositories. In: Muehlen, M.z., Su, J. (eds.) BPM 2010 Workshops. LNBP, vol. 66, pp. 535–546. Springer, Heidelberg (2011)
5. Messmer, B.T.: Efficient graph matching algorithms (1995)
6. Murata, T.: Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4), 541–580 (1989)
7. Petri, C.A.: Kommunikation mit automaten (1962)
8. Rosemann, M.: Potential pitfalls of process modeling: part a. *Business Process Management Journal* 12(2), 249–254 (2006)
9. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business process management: A survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
10. Verbeek, H.M.: Verification of wf-nets (2004)
11. Wang, J., He, T., Wen, L., Wu, N., ter Hofstede, A.H.M., Su, J.: A behavioral similarity measure between labeled petri nets based on principal transition sequences. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 394–401. Springer, Heidelberg (2010)
12. Wang, J., Jin, T., Wong, R.K., Wen, L.: Querying business process model repositories. *World Wide Web*, 1–28 (2013)
13. Wang, S., Wen, L., Wei, D., Wang, J., Yan, Z.: Ssdt-matrix based behavioral similarity algorithm for process models. *Computer Integrated Manufacturing System* 19(8), 1822–1831 (2013)
14. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. *IEEE Transactions on Software Engineering* 37(3), 410–429 (2011)
15. Zha, H., Wang, J., Wen, L., Wang, C., Sun, J.: A workflow net similarity measure based on transition adjacency relations. *Computers in Industry* 61(5), 463–471 (2010)

CFS: A Behavioral Similarity Algorithm for Process Models Based on Complete Firing Sequences

Zihe Dong, Lijie Wen, Haowei Huang, and Jianmin Wang

School of Software, Tsinghua University, Beijing 100084, P.R. China
dongzihe1005@163.com,

{wenlj, jmwang}@tsinghua.edu.cn
huanghw10@gmail.com

Abstract. Similarity measurement of process models is an indispensable task in business process management, which is widely used in many scenarios such as organizations merging, user requirements change and model repository management. This paper focuses on a behavioral process similarity algorithm named CFS based on complete firing sequences which are used to express model behavior. We propose a matching theme for two sets of complete firing sequences by A* algorithm with pruning strategy and define new similarity measure method. Experiments show that this method improves rationality than existing behavior-based similarity algorithms.

Keywords: Petri net; Similarity measure, Coverability tree, Complete firing sequences, A* search algorithm.

1 Introduction

In recent years, with the development and application of workflow technology, many large enterprises and public management institutions are using process models to formalize and operate their inner business processes. As the way of business process management getting more and more developed, a large number of models are accumulated. For example, the total number of process models in OA System of CMCC¹ has been over 8,000. Business process management technology combines information technology with management science, and puts them into the use of operational business processes [1]. This is of great significance to decision change, reform and innovation, and performance improvement of these companies and organizations. Similarity measurement of process models is an indispensable task in business process management [2], which is widely used in many scenarios such as organization merging, user requirements changing, and model repository management.

Nowadays, many process similarity algorithms are proposed to measure the similarity of two process models, and they are based on three different aspects: i) task labels, events and other modeling elements; ii) the topologic structure of the models;

¹ <http://www.10086.cn/>

iii) the execution semantics of the models (i.e. behavior) [3]. In this paper, we focus on the similarity motivated by principal transition sequences (PTS) [4] and propose an improvement scheme by defining the complete firing sequences to express model behavior and defining new similarity measure. We name this new algorithm CFS.

A behavioral algorithm named PTS measuring the similarity of two labeled Petri nets was proposed by Wang [4]. PTS divides a Petri net's behavior into three kinds of principal transition sequences, which includes limited loops, infinite loops and those without loops. By calculating the similarity of each kind of transition sequences and adding them up by weight, we can get the precise similarity of two Petri nets. However, this algorithm considers the partial sub-structures with loops and those without loops in models separately, which breaks the completeness of the execution semantics. As a result, PTS cannot handle Petri nets with loops effectively. Furthermore, because of the absence of some necessary factors such as set-size difference and average similarity between a sequence and a set of sequences in getting the similarity of principal transition sequences, the result of PTS may be unconvincing.

Another method named TAR defines transition adjacency relation (TAR for short) as the gene of a Petri net's behavior and measures the similarity between two sets of TARs using Jaccard coefficient [5]. TAR solves the problem in measuring the similarity of two Petri nets with loop structures. However, it cannot deal with non-free-choice constructs.

The BP method defines strict order relation, exclusiveness relation and interleaving order relation among transitions as the behavioral profiles of a model, and measures the similarity by adding the similarities between these relations up with weights [6]. BP takes into account the multiple sequence relationship between activities and measures the similarity of Petri nets within the time cost of $O(n^3)$. However, the BP method cannot deal with invisible tasks effectively and cannot distinguish interleaving order relations induced from a parallel structure or a loop structure.

The SSDT method [7] computes the shortest succession distance between every two tasks in the model to generate a two-dimensional matrix. Two matrixes after dimension formatting can be used to calculate similarity. It saves the intact behavior characteristics of the model and runs fast. However, the matrix cannot be used as a kind of model index to accelerate process retrieval.

CF algorithm [8] translates process models into causal footprints which contain activity sets, look-back links and look-ahead links. Thus the CF method can express models as the points in spatial vector. It gets similarity by calculating the angle cosine of vectors. However, it cannot distinguish XOR, AND, and OR connectors.

In this paper we propose a new process similarity algorithm based on the complete firing sequences derived from the coverability tree of a labeled Petri net, namely CFS algorithm. This algorithm provides us with more convincing results in comparison with other algorithms in the experiments which we will discuss later in this paper.

The major contributions of this paper are as follows:

1. We manage to express the behavior of a process model by defining complete firing sequences, and propose a method to generate complete firing sequences sets by traversing the coverability tree.

2. Considering many properties, the algorithm we put forward can describe the similarity between two sets of complete firing sequences in a more comprehensive way. A* searching algorithm with effective pruning strategy is used to get the best match of complete firing sequences.
3. We compare this method with PTS and other popular similarity algorithms with detailed experiments and raise a new property that a good process similarity method should meet.

The remainder of this paper is structured as follows. Section 2 presents related concepts. Section 3 introduces CFS method in detail. Section 4 evaluates CFS and compares it with other algorithms. Section 5 summarizes the work and sketches some future work.

2 Preliminaries

In this section we introduce the definitions about Petri net and coverability tree related to CFS which include Petri net system, labeled Petri net system, firing sequence, complete firing sequence and coverability tree.

2.1 Petri Net

Definition 1 (Petri net system [9]): A Petri net system is a quad-tuple $\Sigma = (P, T, F, M_0)$:

- P is a finite set of places;
- T is a finite set of transitions such that $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$;
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of flow relations;
- $M_0 : P \rightarrow \mathbb{N}$, where \mathbb{N} is the set of non-negative numbers.

The algorithm proposed in this paper is based on labeled Petri net, and a labeling function is used to assign names to transitions, which maps transitions to real business activities associated with them.

Definition 2 (Labeled Petri net system [4]): A labeled Petri net system is a sextuple $\Sigma = (P, T, F, M_0, A, L)$, and the quad-tuple (P, T, F, M_0) represents the Petri net system. The others are defined as follows:

- A is a finite set of activity names;
- L is the mapping from transition set to $A \cup \{\varepsilon\}$, ε is the empty label of an invisible activity or a silent activity.

In this paper we measure the similarity of Petri nets for real business processes which are always satisfying the definition of Workflow net.

Definition 3 (Workflow net [10]): A Petri net system $\Sigma = (P, T, F, M_0)$ is a workflow net (WF-net for short) with two special places i and o if it satisfies:

- $i \in P$ is the unique source place such that $\cdot i = \emptyset$;
- $o \in P$ is the unique sink place such that $o \cdot = \emptyset$;
- If we add a transition t^* satisfying $\cdot t^* = \{o\}$ and $t^* \cdot = \{i\}$, the Petri net we get is strongly connected.

Definition 4 (Firing sequence [11]): For a given workflow net $\Sigma = (P, T, F, M_0)$ with the source place i and the sink place o . Let $t \in T$ and M_1 be a marking of the workflow net. If t is enabled at M_1 , it means that for any $p \in t$, $M_1(p) \geq 1$:

- $M_1[t > M_2]$ indicates that t is triggered at M_1 , and the marking is changed to M_2 . Namely, for any $p \in t \setminus t \bullet$, $M_2(p) = M_1(p) - 1$, for any $p \in t \bullet \setminus t$, $M_2(p) = M_1(p) + 1$, and for any other place, $M_2(p) = M_1(p)$;
- $\sigma = < t_1, t_2, \dots, t_n > \in T^*$ is a firing sequence from state M_1' to M_{n+1}' , indicated by $M_1'[\sigma > M_{n+1}']$ if there exist markings M_2', \dots, M_n' such that $M_1'[t_1 > M_2'[t_2 > \dots M_n'[t_n > M_{n+1}']$.

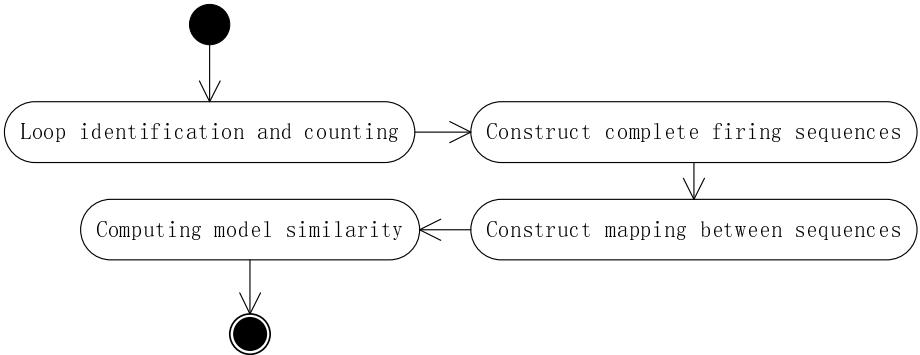
Definition 5 (Complete firing sequence [11]): For a given workflow net $\Sigma = (P, T, F, M_0)$ with the source place i and the sink place o , let $\sigma \in T^*$. Let M_i be the initial marking and $M_i(i) = 1$, and $M_i(p) = 0$ if $p \neq i$. Let M_o be the ending (or sink) marking and $M_o(o) = 1$, and $M_o(p) = 0$ if $p \neq o$. If $M_i[\sigma > M_o]$, σ is defined as a complete firing sequence.

2.2 Coverability Tree

This paper uses coverability tree to analyze the behavior of a Petri net. The procedure of coverability tree generation is essentially a procedure which repeatedly iterates all the markings that can be reached, and the last result is a tree with the markings as nodes and the firings of transitions as edges. The markings can be reached will be countless if the Petri net has no bounds, which makes the coverability tree endless. In order to resolve this problem, a special symbol ω is introduced to represent the situation that a place may have unlimited tokens. For any positive integer n , it satisfies that $\omega > n$, $\omega \pm n = \omega$ and $\omega \geq \omega$. The algorithm of coverability tree generation is illustrated in [12].

3 The CFS Algorithm

The procedure of CFS algorithm is shown in Figure 1. First, we construct the set of complete firing sequences on coverability tree, which includes loop identification, counting method and construction method for complete firing sequences. And then, we calculate model similarity with complete firing sequences including the application of A* algorithm to construct an optimal match between two sets of complete firing sequences and the similarity formula.

**Fig. 1.** The procedure of CFS algorithm

3.1 Constructing the Set of Complete Firing Sequences

The set of complete firing sequences is infinite in the model with loop structures and cannot be used to calculate similarity directly. This section presents a method to construct the limited set of complete firing sequences by specifying the loop number. First we define loop in a Petri net and its corresponding entrance place.

Definition 6 (Loop): Let $\Sigma = (P, T, F, M_0)$ be a Petri net. Loop $N_C = (x_1, \dots, x_n) \in (P \cup T)^*$ satisfies:

- $n > 2$ and for any $1 \leq i, j < n$ with $i \neq j$, $x_i \neq x_j$;
- $(x_i, x_{i+1}) \in F$ for any integer i where $1 \leq i < n$;
- $x_1 = x_n$.

Definition 7 (Entrance places of a loop): Let Petri net $\Sigma = (P, T, F, M_0)$ be a WF-net and one of its loops be $N_C = (x_1, \dots, x_n)$, an entrance place $p_e \in P$ of N_C satisfies:

- There is an integer i ($1 \leq i \leq n$), such that $p_e = x_i$;
- There is a transition $t \in T$ such that $p_e \in t \cdot$ and for any integer i where $1 \leq i \leq n$, we have $t \neq x_i$;

Entrance places are used to identify loop structures. We consider that several loop structures belong to one equivalent loop class if they have the same entrance place. We define equivalent loop class as follows:

Definition 8 (Equivalent loop class): Let Petri net $\Sigma = (P, T, F, M_0)$ be a workflow net, if loop N_1, N_2, \dots, N_n have the same entrance place, they belong to one equivalent loop class. If there is not any other loop which has this entrance place, the equivalent loop class C can be expressed as $C = \{N_1, N_2, \dots, N_n\}$.

We will find equivalent loop class in the coverability tree to express loop unrolling in a model's behavior.

There are three kinds of special nodes in a coverability tree, i.e., anchor, old, dead-end [12]. In the coverability tree, the loop structure is the path from the anchor node to the old node and we define this kind of path as follows:

Definition 9 (Loop subpath, Loop marking): Let $\Sigma = (P, T, F, M_0)$ be a Petri net and its corresponding coverability tree be CT_Σ , in CT_Σ each old node has only one corresponding anchor node, and the path from the anchor node to the old node expresses the loop structure in the Petri net. This path is called loop subpath, which is expressed by $\text{anchorMarking} \xrightarrow{t_1 \dots t_n} \text{oldMarking}$ (for any $1 \leq i \leq n$, t_i is on the path from the anchor node to the old node), and oldMarking is called loop marking, expressed by m_p .

According to the definition of coverability tree, the equivalent loop class in the tree is the set of loop subpaths with the same loop marking. So in the coverability tree, we use $C_{\text{loopmarking}}$ to express an equivalent loop class. We propose a method to compute the execution times of an equivalent loop class in a coverability tree.

Let $p: m_p \xrightarrow{t_1 t_2 \dots t_i} m_p (i \geq 1)$ be a loop subpath in a coverability tree and current firing sequence is σ by traversing the tree. If $t_1 t_2 \dots t_i$ is the discontinuous subsequence of σ , we consider the traversal going through the loop subpath in the generation of σ . To calculate the times of going through the loop subpath p , we use t_{p_end} to express the last transition of p . If the traversal goes through t_{p_end} once, the traversal goes through p for once. Use $\text{path_count}(p)$ to express the times of going through p and $\text{edge_count}(t_{p_end})$ to express the times of going through t_{p_end} , then we have

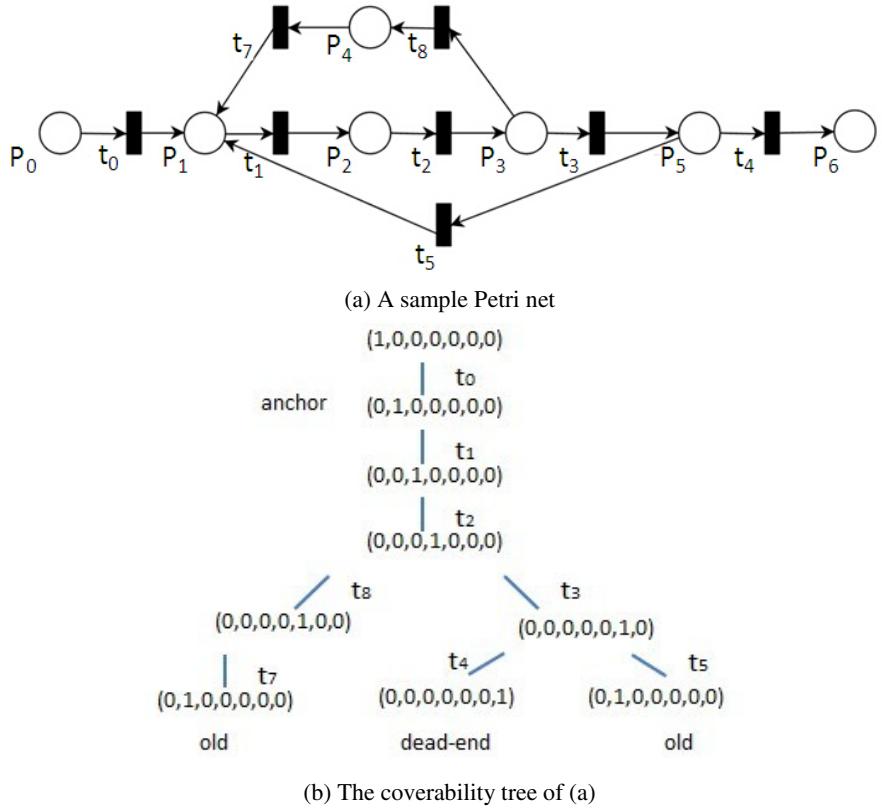
$$\text{path_count}(p) = \text{edge_count}(t_{p_end}) \quad (1)$$

Assume P is the set of loop subpaths in the coverability tree and $\text{path_count}(p)$ is the times of going through loop subpath p in the current firing sequence σ , then the times of C_{marking} in σ expressed by $\text{cycle_count}(C_{\text{marking}})$ is defined as follows:

$$\text{cycle_count}(C_{\text{marking}}) = \sum_{p \in P, m_p = \text{marking}} \text{path_count}(p) \quad (2)$$

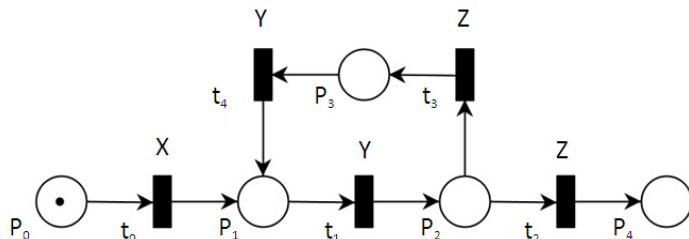
For the Petri net shown in Figure 2(a), the equivalent loop class is $C = \{(P_1, t_1, P_2, t_2, P_3, t_8, P_4, t_7, P_1), (P_1, t_1, P_2, t_2, P_3, t_3, P_5, t_5, P_1)\}$. Correspondingly in the coverability tree of Figure 2(b), the equivalent loop class $C_{(0,1,0,0,0,0,0)}$ includes the following two loop subpaths $p_1: (0,1,0,0,0,0,0) \xrightarrow{t_1 t_2 t_8 t_7} (0,1,0,0,0,0,0)$ and $p_2: (0,1,0,0,0,0,0) \xrightarrow{t_1 t_2 t_3 t_5} (0,1,0,0,0,0,0)$. If the current firing sequence is $t_0 t_1 t_2 t_3 t_5 t_1 t_2 t_8 t_7 t_1 t_2 t_8 t_7 t_1 t_2$, the execution times of the equivalent loop class $\text{cycle_count}(C_{(0,1,0,0,0,0,0)}) = \text{path_count}(p_1) + \text{path_count}(p_2) = \text{edge_count}(t_5) + \text{edge_count}(t_5) = 2 + 1 = 3$.

When constructing complete firing sequences by traversing a coverability tree, we use user-defined parameter k to limit the max execution times of an equivalent loop class. For each firing sequence, if the execution times of some equivalent loop class exceeds k , this firing sequence will be discarded, otherwise continue to expand. We get a complete firing sequence when the execution times of each equivalent loop class is not bigger than k .

**Fig. 2.** A sample Petri net and its coverability tree

3.2 Similarity Algorithm Based on Two Sets of Complete Firing Sequences

According to the definition of labeled Petri net, each transition has been assigned an activity name, namely a label. Let $L(\sigma)$ be the labeled sequence determined by σ , then the set of labeled sequences can be derived from the set of complete firing sequences directly. For instance, one complete firing sequence of the Petri net in Figure 3 is $t_0t_1t_3t_4t_1t_2$, and the corresponding labeled sequence $L(t_0t_1t_3t_4t_1t_2)$ is XYZYYZ.

**Fig. 3.** A sample labeled Petri net $L\Sigma_1$

Firstly, we compare two complete firing sequences by longest common subsequence method. Let σ and σ' be two firing sequences, and $\text{lcs}(L(\sigma), L(\sigma'))$ be the longest common subsequence, and then the similarity of the two firing sequences is:

$$\text{sims}(\sigma, \sigma') = \frac{\text{length}(\text{lcs}(L(\sigma), L(\sigma')))}{\max(\text{length}(L(\sigma)), \text{length}(L(\sigma')))} \quad (3)$$

The similarity between two sets of complete firing sequences is defined as follows. It takes the similarity between the elements and radix distance into consideration and defines the mapping between the two sets, and applies average similarity to compare the complete firing sequences which cannot be mapped with the other set.

Let A and B be two sets of complete firing sequences, assume $|A| \leq |B|$, $\text{avg}(A, \sigma')$ represents the similarity between the set of complete firing sequences A and a complete firing sequence σ' , then we have

$$\text{avg}(A, \sigma') = \frac{\sum_{\sigma \in A} \text{sims}(\sigma, \sigma')}{|A|} \quad (4)$$

Let $\text{dis}(A, B)$ be radix distance between two sets of complete firing sequences A and B where $|A| \leq |B|$, then we have

$$\text{dis}(A, B) = 1 - \frac{|A|}{|B|} \quad (5)$$

Let $M: A \rightarrow B$ be an injective mapping, that maps the complete firing sequences in A to the complete firing sequences in B , $\text{cod}(M)$ be codomain of M , and $B' = B - \text{cod}(M)$, $n \geq 1$, then the similarity between A and B is defined as follows. In the formula, different user-defined parameter n which is an integer decides the influence degree of $\text{dis}(A, B)$ to $\text{simc}(A, B)$.

$$\text{simc}(A, B) = \frac{\sum_{(\sigma, \sigma') \in M} \text{sims}(\sigma, \sigma') + \sum_{\sigma' \in B'} \text{avg}(A, \sigma')}{|B|} \times \left(1 - \frac{\text{dis}(A, B)}{n}\right) \quad (6)$$

CFS uses the set of complete firing sequences to describe the behavior of a labeled Petri net, then the similarity of two labeled Petri nets can be transferred to the similarity between two sets of complete firing sequences. Let $L\Sigma_A$ and $L\Sigma_B$ be the two labeled Petri nets and the corresponding set of complete firing sequences be A and B . Assume $|A| \leq |B|$, then the similarity of the two Petri nets is:

$$\text{Sim}(L\Sigma_A, L\Sigma_B) = \text{simc}(A, B) \quad (7)$$

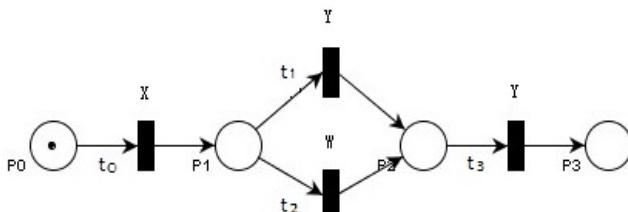


Fig. 4. Another labeled Petri net $L\Sigma_2$

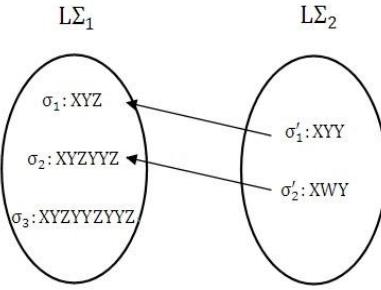


Fig. 5. Injective mapping between $L\Sigma_1$ and $L\Sigma_2$

Take the labeled Petri nets in Figure 3 and Figure 4 as an example. Let upper bound of the execution times of any equivalent loop class be 2, then we can get the sets of complete firing sequences of the two models respectively. We assume the injective mapping between the two sets as Figure 5 indicates, namely $M = \{(\sigma'_1, \sigma_1), (\sigma'_2, \sigma_2)\}$, then we have $B' = \{\sigma_3\}$, let n in Equation 6 be 10^3 , then

$$\text{Sim}(L\Sigma_1, L\Sigma_2) = \frac{\frac{2}{3} + \frac{1}{3} + \frac{1}{2}(\frac{1}{3} + \frac{2}{9})}{3} \times \left(1 - \frac{\frac{1}{3}}{10^3}\right) = 0.426$$

3.3 Optimal Mapping between Two Sets of Complete Firing Sequences

We can get different similarity when applying different mapping plans between the two sets of complete firing sequences in the similarity calculation of two labeled Petri nets. We define the similarity of two labeled Petri nets to be the maximum value of all the possible similarities under different mapping plans, and the corresponding mapping is called the optimal mapping.

From Equation 6, we can infer that the maximum value of $\sum_{(\sigma, \sigma') \in M} \text{sim}(\sigma, \sigma') + \sum_{\sigma' \in B'} \text{avg}(A, \sigma')$ can achieve the maximum value of the similarity between two Petri nets, so we need to find the optimal mapping plan which makes the maximum value. The time complexity of brute-force method is $O(n!)$, and the result calculated by greedy algorithm is always not the global optimum. Therefore, we use A^* algorithm to construct the optimal mapping.

The construction of the optimal mapping can be treated as the construction of the mapping tree. As Figure 6 shows, each node represents an intermediate result (M, A', B') , M is a partial mapping currently, A' is the unmapped sequences in the set of complete firing sequences A and B' is the unmapped sequences in the set of complete firing sequences B . In the handing procedure of the algorithm, for each leaf node, evaluation function $f(M)$ contains two parts:

$$f(M) = g(M) + h(M) \quad (8)$$

In Equation 8, $g(M)$ implies the solution of the current partial mapping M , $h(M)$ indicates the upper bound of the extended solution of unmapped sequences, so $f(M)$ is the upper bound of all the mapping solution extended from M . The algorithm extends the leaf node with the maximum $f(M)$ each time, and it ends when the leaf node contains a complete mapping, namely the optimal solution. In this procedure only partial nodes can be extended to implement pruning and improve efficiency. The pseudo code of the algorithm is as follows.

```

input: two sets of complete firing sequences A, B, and
|A| ≤ |B|
output:  $f(M)$  corresponding to the optimal mapping  $M$ 
AStarAlgorithmforMap(A, B)
open:= $\{(\sigma, \sigma')\} | \sigma \in A, \sigma' \in B\}$ 
while open ≠  $\emptyset$  do
    begin
         $M := \arg \max_{M' \in \text{open}} g(M') + h(M')$ 
        open:=open \ {M}
        if dom( $M$ ) = A then return  $g(M) + h(M)$ 
        else
            begin
                select  $\sigma \in A$ , such that  $\sigma \notin \text{dom}(M)$ 
                foreach  $\sigma' \in B$ , such that  $\sigma' \notin \text{cod}(M)$ 
                    begin
                         $M'' := M \cup \{(\sigma, \sigma')\}$ 
                        open:=open ∪ { $M''$ }
                    end
            end
    end
end

```

This algorithm firstly initializes the mapping collection $open$ which represents the collection comprised by current leaf nodes. In the procedure of loop, the algorithm extends the mapping with the maximum $g(M) + h(M)$, and excludes M from $open$. M is the optimal mapping, if its definition domain equals to A , and the algorithm returns with $g(M) + h(M)$. Otherwise, the algorithm chooses an unmapped sequence σ from A , and for each unmapped sequence σ' in B , it constructs a mapping pair (σ, σ') and adds the new mapping pair to the new mapping M'' derived from M , and M'' is added to $open$.

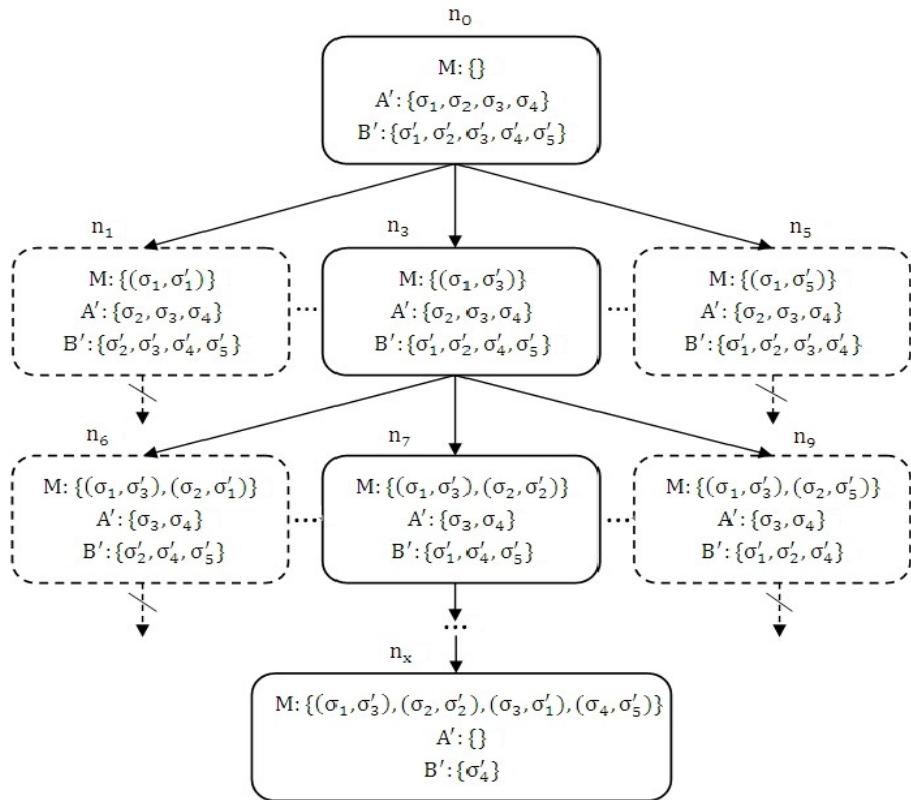


Fig. 6. The execution procedure of A* algorithm in tree structure

Here gives the definition of $g(M)$ and $h(M)$ which is used when pruning:

A^* algorithm calculates the optimal mapping, and finally return the maximum of $\sum_{(\sigma, \sigma') \in M} \text{sims}(\sigma, \sigma') + \sum_{\sigma' \in B'} \text{avg}(A, \sigma')$ in Equation 6, then the evaluation function $f(M') = g(M') + h(M')$ is the upper bound of $\sum_{(\sigma, \sigma') \in M} \text{sims}(\sigma, \sigma') + \sum_{\sigma' \in B'} \text{avg}(A, \sigma')$ corresponding with all the possible mapping M extended from M' . $g(M)$ is the current partial solution, and $h(M)$ is the upper bound of all solutions of partial mappings that will be extended.

Assume that the mapping of current node is M , then the set of unmapped complete firing sequences in A is defined as $A' = A - \text{dom}(M)$, and the set of unmapped complete firing sequences in B is defined as $B' = B - \text{cod}(M)$. Assume $|A| \leq |B|$, then the definition of $g(M)$ is as follows.

$$g(M) = \begin{cases} \sum_{(\sigma, \sigma') \in M} \text{sims}(\sigma, \sigma'), & A' \neq \emptyset \\ \sum_{(\sigma, \sigma') \in M} \text{sims}(\sigma, \sigma') + \sum_{\sigma' \in B'} \text{avg}(A, \sigma'), & A' = \emptyset \end{cases} \quad (9)$$

The value of $g(M)$ is the similarity of current partial mapping, which is the determined part. When $A' \neq \emptyset$ and there exists unmapped sequences in A , $g(M)$ is the sum of similarity of each sequence pair. If $A' = \emptyset$ and all the sequences have been mapped, M will be the complete solution with all the values having been determined, then $g(M)$ is equal to $\sum_{(\sigma, \sigma') \in M} \text{sims}(\sigma, \sigma') + \sum_{\sigma' \in B'} \text{avg}(A, \sigma')$ corresponding with M .

Assume that the mapping of the current node is M , then the set of unmapped complete firing sequences in A is $A' = A - \text{dom}(M)$ and the set of unmapped complete firing sequences in B is $B' = B - \text{cod}(M)$. If $|A| \leq |B|$, let $S = \{B'' | B'' \subseteq B'\}$, and $|B''| = |B| - |A|\}$, then the definition of $h(M)$ is as follows.

$$h(M) = \begin{cases} \sum_{\sigma \in A'} \max_{\sigma' \in B'} \text{sims}(\sigma, \sigma') + \max_{B'' \in S} \sum_{\sigma'' \in B''} \text{avg}(A, \sigma''), & A' \neq \emptyset \\ 0, & A' = \emptyset \end{cases} \quad (10)$$

The $h(M)$ is comprised of two parts, if $A' \neq \emptyset$ among which the first part is the sum of the maximum similarity between each element in A' and some element in B' , the second part is sum of the similarity between each element in B'' and the set A , in which B'' is a subset of B' and its size is $|B| - |A|$. If there exists many possible B'' , the B'' having the maximum sum will be chosen. If $A' = \emptyset$ and all the sequences in A have been mapped and no other sequence needs to be extended, M is the complete solution with no evaluation part, and $h(M) = 0$ obviously.

The paper by Peter E. Hart [13] has proven the admissibility of A^* searching algorithm, and insures that A^* search algorithm can give the optimal solution, the admissibility in this problem is defined as follows.

Theorem 1. For any partial mapping M generated by A^* search algorithm, let S be the collection comprised of all the complete solutions extended by M . Let $M' = \arg \max_{M \in S} g(M)$, then A^* is admissible if $h(M) \geq g(M') - g(M)$, namely $h(M)$ is the upper bound of the partial solution getting from mapping remaining sequences.

Proof: Let the complete solution of A^* algorithm be an injective mapping $A \rightarrow B$. For any partial mapping M , we have $A' = A - \text{dom}(M)$ and $B' = B - \text{cod}(M)$.

If $A' \neq \emptyset$, let $M_{\text{new}} = M' - M$ and $B_{\text{un}} = B - \text{cod}(M')$ which represents the set of unmapped firing sequences eventually. Then we have $g(M') - g(M) = \sum_{(\sigma, \sigma') \in M_{\text{new}}} \text{sims}(\sigma, \sigma') + \sum_{\sigma'' \in B_{\text{un}}} \text{avg}(A, \sigma'')$. Since $\text{dom}(M_{\text{new}}) = A'$ and $\text{cod}(M_{\text{new}}) \subseteq B'$, $\sum_{\sigma \in A'} \max_{\sigma' \in B'} \text{sims}(\sigma, \sigma') \geq \sum_{(\sigma, \sigma') \in M_{\text{new}}} \text{sims}(\sigma, \sigma')$. Let $S = \{B'' | B'' \subseteq B', |B''| = |B| - |A|\}$, then $B_{\text{un}} \in S$, so we have $\max_{B'' \in S} \sum_{\sigma'' \in B''} \text{avg}(A, \sigma'') \geq \sum_{\sigma'' \in B_{\text{un}}} \text{avg}(A, \sigma'')$. Therefore, $h(M) = \sum_{\sigma \in A'} \max_{\sigma' \in B'} \text{sims}(\sigma, \sigma') + \max_{B'' \in S} \sum_{\sigma'' \in B''} \text{avg}(A, \sigma'') \geq \sum_{(\sigma, \sigma') \in M_{\text{new}}} \text{sims}(\sigma, \sigma') + \sum_{\sigma'' \in B_{\text{un}}} \text{avg}(A, \sigma'') = g(M') - g(M)$.

The partial mapping M is the complete solution if $A' = \emptyset$, namely $M = M'$. In this condition, $h(M) = 0$ and $g(M') = g(M)$, so $h(M) = g(M') - g(M) = 0$ which satisfies $h(M) \geq g(M') - g(M)$.

4 Experimental Evaluation

By measuring the satisfaction rate of the triangle inequality in real process models, we noticed that when we set the upper bound k to 2 and n in the similarity formula to 1000, we can get the highest satisfaction rate of the triangle inequality. We utilize these recommended values to investigate the pruning strategy in A* and to compare CFS with popular similarity algorithms. Our model set contains 124 models from TC, 115 models from DG, 592 reference models from SAP and 200 models generated by BeehiveZ [14]. Our experiment environment is Eclipse Platform whose version is 3.6.1 and JDK is 1.7.0_25 with 4G memory assigned. And the computer has OS of Red Hat Enterprise Linux Server (Santiago) and the CPU is Intel(R) Xeon(R) CPU E5-2640 whose main frequency is 2.50GHz.

4.1 Experiments on Pruning Strategy of A* Algorithm

From the space point, we discuss the rate of model pairs calculated before and after applying pruning strategy, which is the ratio of calculated model pairs vs. total pairs. Experiment result is shown in Figure 7. If pruning strategy is not applied, the rate reaches 69.26% at most which results from the space complexity of factorial level. Otherwise, the rate reaches 100% because pruning strategy only extends the partial mapping which may generate the optimal mapping and saves the storage space.

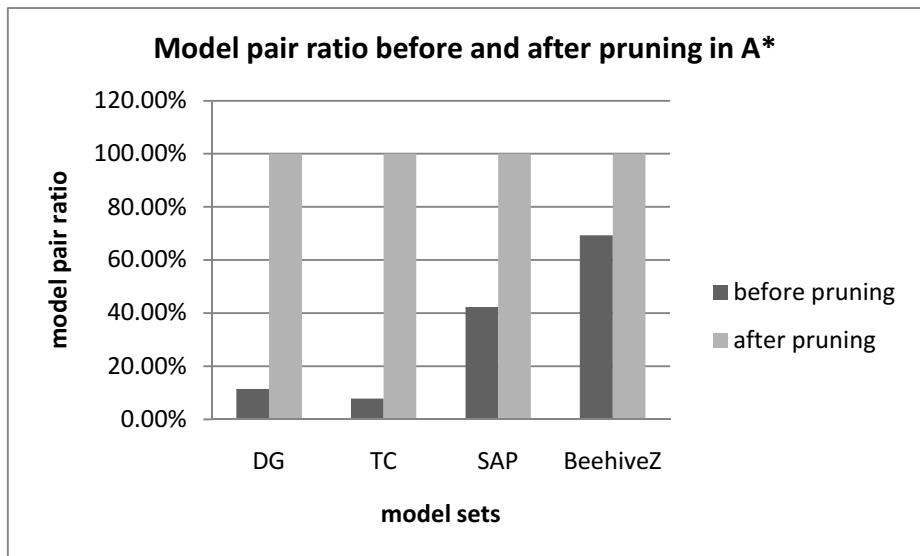


Fig. 7. Model pair ratio before and after pruning in A*

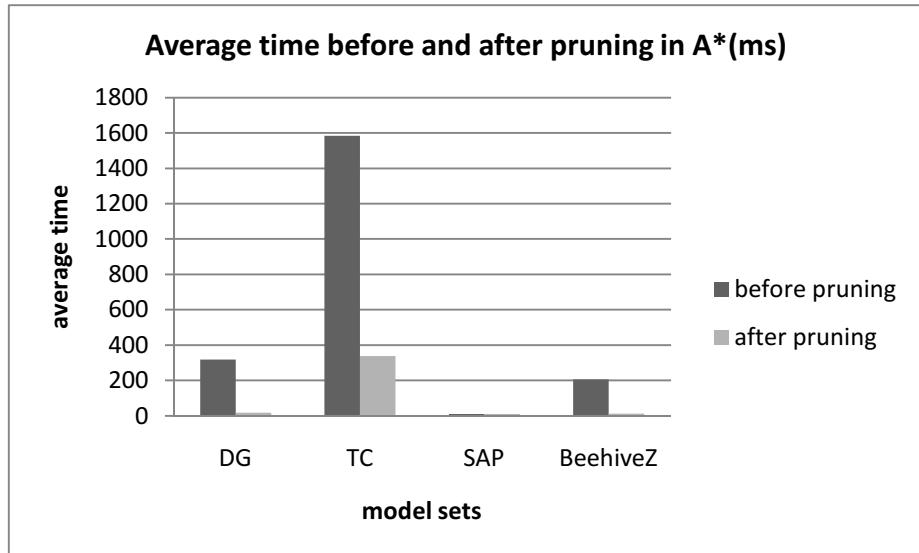


Fig. 8. Average time before and after pruning in A*(ms)

From the time point, we measure average calculation cost based on the calculated model pairs before and after applying pruning strategy. The result is shown in Figure 8, from which we can infer that applying pruning strategy significantly reduces the execution time.

4.2 Compared with Other Similarity Algorithms Based on Model Behavior

Wang et al. give the five properties that process similarity algorithms should satisfy in [7]. These properties coincide with real business and human intuition, and can be criterions of measuring similarity algorithms in some degree. We have compared CFS algorithm with other behavioral process similarity algorithms by the five properties, and the result is shown in Table 1.

Table 1. Evaluation result of similarity algorithms on the five properties

properties	SSDT	PTS	TAR	CF	BP	CFS
Property1	√	√			√	√
Property2	√	√				√
Property3	√		√	√	√	√
Property4	√				√	√
Property5	√	√		√	√	√

From Table 1, we can conclude that PTS, TAR, CF, BP all have unsatisfied properties and only SSDT and CFS can satisfy all the five properties. On the following new property we compare these algorithms further.

We propose a new property based on the analysis of real business process models:

Let Σ_1 and Σ_2 be two business process models, and both of them are referred to two kinds (A and B) of complete firing sequence sets. Assume A and B are two different kinds of businesses between which the similarity of their complete firing sequence sets is very low. However, the similarities between the same kinds of sequences are very high. In sequences of Σ_1 , A is the majority and B is in the minority, and they are opposite in model Σ_2 . After mapping the complete firing sequences between these two models, few mapping pairs have high similarities and most of the mapped pairs have the low values, then we think the similarity between the two models is low. The different businesses have different proportions in the two models and we call this property as unbalance of business behavior distribution.

Take the models Σ_1 and Σ_2 in Figure 9 as an example, the two sets of complete firing sequences and the mapping between them can be figured out in Figure 10. The similarity of the two models is 0.286 by our algorithm, and the similarities by other algorithms are listed in Table 2. Based on the results, we conclude that the result of TAR and CFS conform to our expectation, while the results of the other algorithms are too much higher than our expectation.

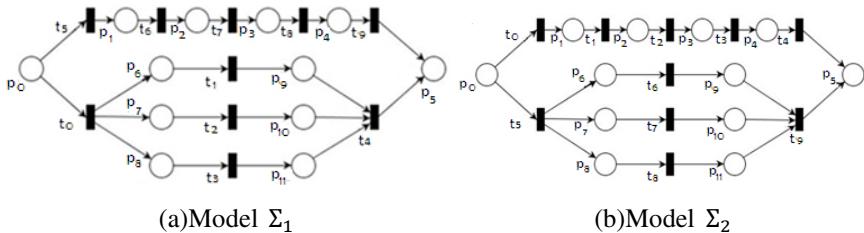


Fig. 9. Two example models for the new property

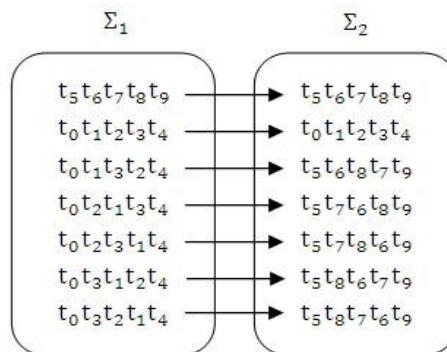


Fig. 10. The optimal mapping between the two models in Figure 9

Table 2. The similarity of the two models in Figure 9 by different algorithms

Algorithm	Similarity between Σ_1 and Σ_2
SSDT	0.840
PTS	0.829
TAR	0.333
CF	0.858
BP	0.680
CFS	0.286

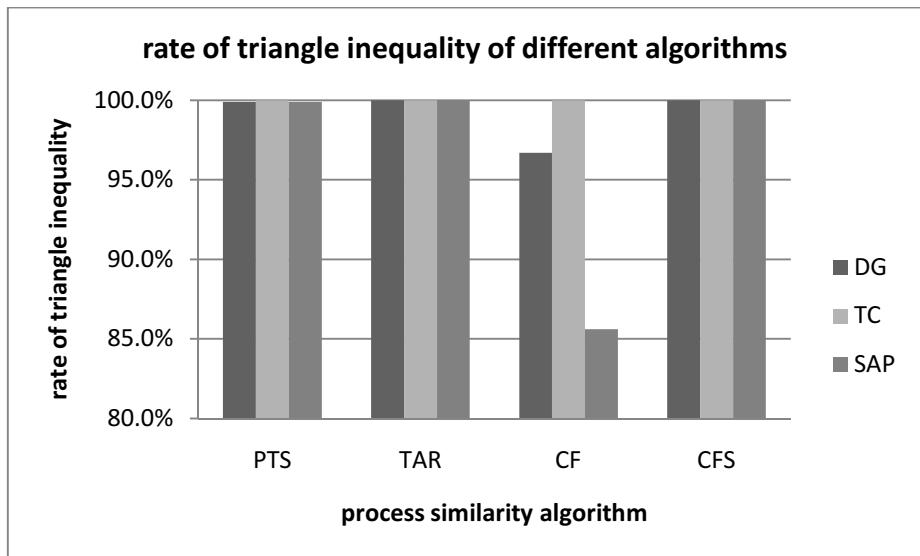
Here we discuss the satisfaction rate of triangle inequality of PTS, TAR, CF and CFS and set the threshold of similarity calculation time to five minutes. Matthias [15] gives the definition of triangle inequality. Let S be the model collection, and $|S| = n$. Let any three of the models make a group, then we can get C_n^3 groups. If m of the C_n^3 groups satisfy triangle inequality, the satisfaction rate of triangle inequality of set S is defined as follows.

$$\text{Rate}(S) = \frac{m}{C_n^3} \quad (11)$$

Let $L\Sigma_A$ and $L\Sigma_B$ be two labeled Petri nets, then we can transfer the similarity to distance using following equation.

$$\text{Dis}(L\Sigma_A, L\Sigma_B) = 1 - \text{Sim}(L\Sigma_A, L\Sigma_B) \quad (12)$$

The satisfaction rates of triangle inequality by different algorithms are shown in Figure 11. TAR and CFS can satisfy the triangle inequality completely, and PTS does slightly less than 100% while CF cannot satisfy the triangle inequality well enough.

**Fig. 11.** The rate of triangle inequality

In conclusion, CFS has advantages over other algorithms on the six properties, while TAR, PTS, CF, BP and SSDT all have some unsatisfied properties. Furthermore, CFS is more reasonable and intuitional. As to the rate of triangle inequality, CFS's satisfaction rate reaches 100% and is better than PTS and CF.

5 Summary and Future Work

In this paper we propose a new process model similarity algorithm named CFS based on labeled Petri net and coverability tree. It expresses the behavior of process models by the set of complete firing sequences, and deals with all kinds of structures in Petri net efficiently, especially loops. Also, we improved the execution efficiency of the A* search algorithm using pruning strategy. As we can see from the experimental results, CFS is more reasonable than any other process similarity algorithms based on the model behavior.

In future work, we will be devoted to improving the calculation efficiency of CFS algorithm, and we are interested in finding better strategy for pruning. Furthermore, we intend to put forward a more common criteria for evaluating process model similarity algorithms.

Acknowledgement. The research is supported by the MOE-CMCC research foundation project No.MCM20123011 and the special fund for innovation of Shandong, China No.2013CXC30001.

References

1. van der Aalst, W.M.P.: Business process management demystified: A tutorial on models, systems and standards for workflow management. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets. LNCS, vol. 3098, pp. 1–65. Springer, Heidelberg (2004)
2. Becker, M., Laue, R.: A comparative survey of business process similarity measures. Computers in Industry 63(2), 148–167 (2012)
3. Dumas, M., García-Bañuelos, L., Dijkman, R.M.: Similarity Search of Business Process Models. IEEE Computer Society Technical Committee on Data Engineering 32(3), 23–28 (2009)
4. Wang, J., He, T., Wen, L., Wu, N., ter Hofstede, A.H.M., Su, J.: A Behavioral Similarity Measure between Labeled Petri Nets Based on Principal Transition Sequences. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 394–401. Springer, Heidelberg (2010)
5. Zha, H., Wang, J., Wen, L., et al.: A workflow net similarity measure based on transition adjacency relations. Computers in Industry 61(5), 463–471 (2010)
6. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity – A proper metric. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 166–181. Springer, Heidelberg (2011)
7. Wang, S., Wen, L., Wang, J., et al.: SSDT-matrix based behavioral similarity algorithm for process models. In: The Third CBPM (2013)

8. Dijkman, R., Dumas, M., van Dongen, B., et al.: Similarity of business process models: metrics and evaluation. *Information Systems* 36(2), 498–516 (2011)
9. Petri, C.A.: Kommunikation mit automaten [PhD]. Fachbereich Informatik of Universitat Hamburg (1962)
10. Van Der Aalst, W.M.P.: Verification of Workflow Nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
11. Gerke, K., Cardoso, J., Claus, A.: Measuring the compliance of processes with reference models. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009, Part I. LNCS, vol. 5870, pp. 76–93. Springer, Heidelberg (2009)
12. Murata, T.: Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4), 541–580 (1989)
13. Hart, P.E., Nilsson, N.J., Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics* 4(2), 100–107 (1968)
14. Wu, N., Jin, T., Zha, H., et al.: BeehiveZ: an open framework for business process model management. *Journal of Computer Research and Development* 47(z1), 450–454 (2010)
15. Kunze, M., Weske, M.: Metric trees for efficient similarity search in large process model repositories. In: Muehlen, M.z., Su, J. (eds.) BPM 2010 Workshops. Lecture Notes in Business Information Processing, vol. 66, pp. 535–546. Springer, Heidelberg (2011)

Efficient Behavioral-Difference Detection between Business Process Models

Zhiqiang Yan^{1,2}, Yuquan Wang², Lijie Wen², and Jianmin Wang²

¹ Capital University of Economics and Business, Beijing, China

² Tsinghua University, Beijing, China

zhiqiang.yan.1983@gmail.com

Abstract. Recently, business process management plays a more and more important role on the management of organizations. Business process models are used to enhance the efficiency of management and gain the control of (co)operations both within an organization and between business partners. Organizations, especially big enterprises, maintain thousands of process models and process models of the same category are with high similarity. For example, China Mobile Communication Corporation (CMCC) have more than 8000 processes in their office automation systems and the processes of the same business in different subsidiaries are with some variations. Therefore, to analyze and manage these similar process models, techniques are required to detect the differences between them. This paper focuses on detecting behavioral differences between process models. The current technique takes hours to compare the behaviors of two models in worst cases. Therefore, in this paper, an efficient technique is proposed by comparing dependency sets and trace sets of features in process models. Experiments show that the technique can detect behavioral differences between two models within seconds in worst cases, at least four orders of magnitude faster than existing techniques.

1 Introduction

Nowadays, business process management is popular in the management of organizations. More and more organizations describe their operations as business processes to enhance their efficiency. It is common for organizations to have collections of thousands of business processes models. Process model repositories techniques are required to manage such amount of models [14,21]. Among these process models, some may have high similarity with others. For example, the information department of China Mobile Communication Corporation (CMCC) maintains more than 8,000 processes of more than 30 subsidiaries in its office automation systems [9]. Process models for the same business in different subsidiaries are almost the same with some variations to fit in local scenarios. The headquarters would like to have a set of “high-level” models that work in all subsidiaries instead of maintaining all similar models. As such, the number of process models reduces and the headquarters can better manage business processes in subsidiaries. To this end, it is required to have a technique that can

detect the differences between process models of the same business in different subsidiaries.

There are two types of difference-detection techniques for process models. The first one focuses on structures of process models. It basically considers process models as graphs and identifies differences on the syntactic level [13,15,16]. The second one focuses on behaviors of process models. It considers execution traces of process models and identifies differences on the semantic level [3,5]. The technique described in this paper follows the second branch. Fig. 1 shows two similar business process models describing a “receiving goods” process. Both models are in the BPMN notation. Given two similar process models, the technique in this paper identifies the difference patterns between these models. For example, in Fig. 1, activity “*Inbound Delivery Entered*” in *Model a* and activity “*Inbound Delivery Created*” in *Model b* are interchanged activities; F1.1 of *Model a* has an additional dependency, “*Invoice received*”, compared with F2.1 of *Model b*; F1.2 of *Model a* has different conditions compared with F2.2 of *Model b* because of different gateways. The technique in this paper detects these differences between process models efficiently.

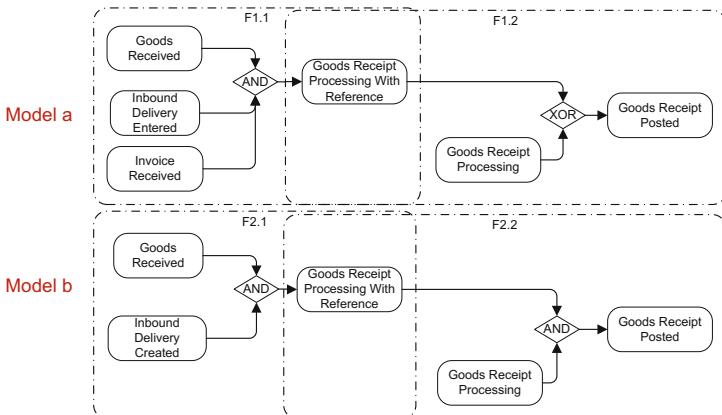


Fig. 1. Two business process models about receiving goods in BPMN

There currently exists a technique [5] that can detect these behavioral differences between process models. However, it uses the notation of completed trace equivalence, which is time consuming in some cases. As shown in its experimental result, the computation of 3% of the model pairs in that experiment cannot finish after three hours.

In this paper, we present a technique that detects these differences between two business process models in milliseconds on average. Given two business process models (in which all the activities can be executed), the technique works in the following steps. Firstly, we divides process models into small features and features refer to most common workflow patterns in this paper, i.e., sequence,

split, join, and loop. Secondly, we match activities between two process models and detects differences related to activities. Lastly, we detect differences by comparing trace sets of these features.

The contribution of this paper is as following:

- experiments show that the technique in this paper detects behavioral differences between two process models in seconds in the worst case, instead of hours in [5];
- each detected difference is precisely related to small features instead of the whole process models;
- a more thorough experiment is run with more than 140,000 process model pairs with high similarity while the experiment in [5] is with 132 pairs.

The rest of the paper is organized as follows. Section 2 defines the concept of feature and presents traces of a feature. Section 3 presents how to detect differences in the context of features. Section 4 presents the steps of detecting differences between two business process models. Section 5 presents the experiments to evaluate the difference detection technique. Section 6 presents related work and section 7 concludes the paper.

2 Features, Feature Traces, Dependencies

This section presents the definition of features and their traces in the context of business process graphs. We define our difference detection technique on business process graphs to be independent of a specific notation (BPMN, EPC, Petri net, etc.) and a similar process graph notation is defined in [7] for the same purpose. During the transformation from a business process model to a business process graph, types of nodes are ignored except for gateways, because the differences in this paper are based on behaviors and gateways are required to analyze behaviors of a business process graph. For example, Fig. 2 shows the business process graphs for the models from Fig. 1.

Definition 1 ((Business) Process Graph, Pre-set, Post-set). Let \mathcal{L} be a set of labels. A process graph is a tuple (N, E, λ) or (A, GW, E, λ) , in which:

- $N = A \cup GW$ is the set of nodes, which consists of a set of activity nodes A and a set of gateway nodes GW and $A \cap GW = \emptyset$;
- $E \subseteq N \times N$ is the set of edges;
- $\lambda : N \rightarrow \mathcal{L}$ is a function that maps nodes to labels.

Let $G = (N, E, \lambda)$ be a process graph and $n \in N$ be a node: $\bullet n = \{m | (m, n) \in E\}$ is the pre-set of n , while $n\bullet = \{m | (n, m) \in E\}$ is the post-set of n .

An activity node can at most has one node in its pre-set (post-set), i.e., $\forall n \in A, |\bullet n| \leq 1 \wedge |n\bullet| \leq 1$. A gateway node has multiple nodes in pre-set or post-set and has a restricted label, more precisely, $\forall n \in GW, (|\bullet n| = 1 \wedge |n\bullet| \geq 2) \vee (|n\bullet| = 1 \wedge |\bullet n| \geq 2) \wedge \lambda(n) \in \{\text{"XOR"}, \text{"OR"}, \text{"AND"}\}$. We call it a split gateway node when it has multiple nodes in its post-set and a join gateway node when it has multiple nodes in its pre-set.

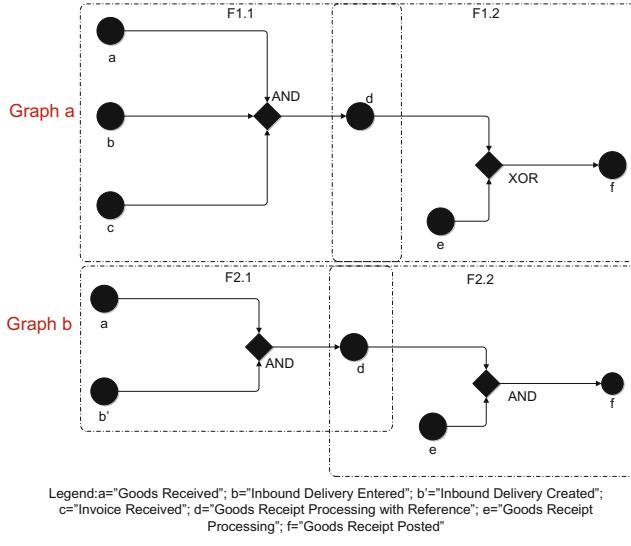


Fig. 2. Business process graphs

Definition 2 (Execution Semantics). Let $g = (A, GW, E, \lambda)$ be a business process graph and $S \subseteq A$ be the state of g . The execution semantics is as follows.

- $\forall a \in S$, a is enabled if $a \bullet \subseteq A$; after the execution, $S = S \cup a \bullet - \{a\}$.
- $\forall gw \in GW \wedge |\bullet gw| = 1$, $\forall a \in \bullet gw$ are enabled if and only if $\bullet gw \subseteq S$; after the execution $S = S \cup A_1 - \bullet gw$, where $A_1 \subseteq gw \bullet$ is determined by $\lambda(gw) \in \{\text{"XOR"}, \text{"OR"}, \text{"AND"}\}$.
- $\forall gw \in GW \wedge |gw \bullet| = 1$, $\forall a \in \bullet gw$ are enabled and can be executed if $\bullet gw \subseteq S$ satisfies the activation conditions of $\lambda(gw) \in \{\text{"XOR"}, \text{"OR"}, \text{"AND"}\}$; after the execution if $gw \bullet \subseteq A$, $S = S \cup gw \bullet - \bullet gw$, otherwise $gw_1 \in gw \bullet$ is enabled and $S = S - \bullet gw$.

We say that S is a proper state if the state becomes an empty set after all possible executions. Let $START = \{\forall a \in A \wedge \bullet a = \emptyset\}$ be the set of start activities in g , and S is a start state if and only if $S \subseteq START$.

For example, let $S = \{a, b, c\}$ be the state of “Graph a” in Fig. 2. Firstly, activities a , b , and c are executed; secondly, the “AND” gateway is activated and $S = \{d\}$; thirdly, activity d is executed; fourthly, the “XOR” gateway is activated and $S = \{f\}$; finally, activity f is executed and $S = \emptyset$.

Definition 3 (Trace (Set)). Given a business process graph $g = (A, GW, E, \lambda)$, Given a proper state S of g , the sequence of executed activities is a trace of g ; the trace set of g consists of possible sequences of executed activities for start states that are also proper states at the same time.

For example, the trace set of “*Graph a*” in Fig. 2 is $\{< a, b, c, d, f >, < a, c, b, d, f >, < b, a, c, d, f >, < b, c, a, d, f >, < c, a, b, d, f >, < c, b, a, d, f >, < e, f >\}$.

We consider features based on the most common workflow patterns: sequence, split, join, and loop. This is because the differences defined in [3] are mostly related to these common workflow patterns. More details about differences will be given in Section 3. Basic features are defined in Definition 4 [22].

Definition 4 (Basic Feature). Let $g = (A, GW, E, \lambda)$ be a process graph. A feature f of g is a subgraph of g . The size of a feature is the number of edges it contains. A feature is a:

- sequence feature of size $s - 1$ consisting of nodes $\{a_1, a_2, a_3, \dots, a_s\} \subset A$, if E_f is the edge set containing $(a_1, a_2), (a_2, a_3), \dots, (a_{s-1}, a_s)$, for $s \geq 2$;
- basic split feature of size $s + 1$ consisting of a split activity node $a \in A$, a split gateway node $gw \in GW$ and a set of nodes $\{a_1, a_2, \dots, a_s\} \subset A$, if and only if E_f is the edge set containing $(a, gw), (gw, a_1), (gw, a_2), \dots, (gw, a_s)$, for $s \geq 2$;
- basic join feature of size $s + 1$ consisting of a join activity node $a \in A$, a join gateway node $gw \in GW$ and a set of nodes $\{a_1, a_2, \dots, a_s\} \subseteq A$, if and only if E_f is the edge set containing $(gw, a), (a_1, gw), (a_2, gw), \dots, (a_s, gw)$, for $s \geq 2$;
- loop feature of size s consisting of nodes $\{n_1, n_2, \dots, n_s\} \subseteq A \cup GW$, if $\exists i, j \in [1, s]$ such that n_i , a join gateway, is the entry of the loop feature and n_j , a split gateway, is the exit of the loop feature; E_f is the edge set containing $(n_1, n_2), \dots, (n_{s-1}, n_s)$, and (n_s, n_1) , for $s \geq 1$.

In Fig.2, features are circled with dash-dotted borders. For example, *F1.1* in *Graph a* is a basic join feature.

The trace set of a basic feature can be computed based on Definition 3. Suppose that there is no “dead” activity in A , i.e., every activity can be executed during executions of the process graph. The trace set of a feature, denoted as T , contains the set of possible traces during its execution.

If the feature is a:

- sequence feature of size s consisting of nodes $\{a_1, a_2, a_3, \dots, a_s\} \subseteq A$ and edges $(a_1, a_2), (a_2, a_3), \dots, (a_{s-1}, a_s) \subseteq E$, the trace set consists of only one trace, i.e., $T = \{< a_1, a_2, a_3, \dots, a_s >\}$.
- basic split feature of size $s + 1$ consisting of a split activity node $a \in A$, a split gateway node $gw \in GW$, a set of nodes $A_s = \{a_1, a_2, \dots, a_s\} \subseteq A$ and a set of edges $\{(a, gw), (gw, a_1), (gw, a_2), \dots, (gw, a_s)\}$, there are three different cases:
 - $\lambda(gw) = \text{“XOR”}$: $T = \{< a, n > | n \in A_s\}$.
 - $\lambda(gw) = \text{“AND”}$: $T = \{< a, t > | t \in A_s^* \wedge |t| = s\}$.
 - $\lambda(gw) = \text{“OR”}$: $T = \{< a, t > | t \in (A_s^* - \emptyset)\}$.
- basic join feature of size $s + 1$ consisting of a join activity node $a \in A$, a join gateway node $gw \in GW$, a set of nodes $A_j = \{a_1, a_2, \dots, a_s\} \subseteq A$, and $\{(gw, a), (a_1, gw), (a_2, gw), \dots, (a_s, gw)\}$, there are three different cases:

- $\lambda(gw)$ = “XOR”: $T = \{< n, a > | n \in A_j\}$.
- $\lambda(gw)$ = “AND”: $T = \{< t, a > | t \in A_j^* \wedge |t| = s\}$.
- $\lambda(gw)$ = “OR”: $T = \{< t, a > | t \in (A_j^* - \emptyset)\}$.
- loop feature of size s consisting of nodes $\{n_1, n_2, \dots, n_s\}$, n_i and n_j are the entry and exit of the loop and edges $\{(n_1, n_2), (n_2, n_3), \dots, (n_{s-1}, n_s), (n_s, n_1)\}$. Let $t \upharpoonright A$ denote the projection of t on the activity set A . The traces of the loop feature is $< n_i, \dots, n_{j-1}, (n_j, n_1, n_2, \dots, n_{j-1})^k > \upharpoonright A$, ($k \geq 0$).

For example, the trace set of *F1.1* is $\{< a, b, c, d >, < a, c, b, d >, < b, a, c, d >, < b, c, a, d >, < c, a, b, d >, < c, b, a, d >\}$.

We also notice that multiple split/join gateways may connect to each other as shown in Fig. 3. Differences occur between fragments like that. For example, the two fragments in Fig. 3 is detected as *different-conditions* because they are with the same set of activities but different behaviors. Therefore, we define complex split/join features to be able to identify differences in complex structures.

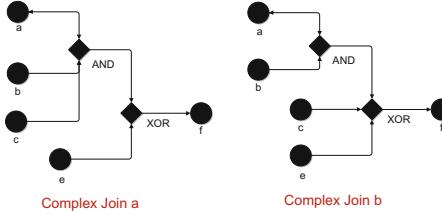


Fig. 3. Complex Join Features

Definition 5 (Complex Feature). Let $g = (A, GW, E, \lambda)$ be a process graph. A subgraph of g $f = (A_1, GW_1, E_1, \lambda)$ is a complex feature if and only if:

- $GW_1 \subseteq GW$ ($|GW_1| > 1$), $\forall gw \in GW_1, \exists gw_1 \in GW_1$, such that $gw_1 \in gw \bullet \forall gw_1 \in \bullet gw$;
- $A_1 \subseteq A$ ($|A_1| > 3$), $\forall a \in A_1, \exists gw \in GW_1$, such that $a \in gw \bullet \forall a \in \bullet gw$;
- $E_1 = E \cap ((A_1 \cup GW_1) \times (A_1 \cup GW_1))$.

We say that f is a complex split feature if and only if $\forall gw \in GW_1, |\bullet gw| = 1 \wedge |gw \bullet| \geq 2$; f is a complex join feature if and only if $\forall gw \in GW_1, |\bullet gw| \geq 2 \wedge |gw \bullet| = 1$.

The trace set of a complex feature can be computed the same way as that of a process graph as defined in Definition 3. For example, the trace set of *Complex Join a* in Fig. 3 is $\{< a, b, c, f >, < a, c, b, f >, < b, a, c, f >, < b, c, a, f >, < c, a, b, f >, < c, b, a, f >, < e, f >\}$.

Besides feature traces, the dependency set of an activity in features are also required to identify behavioral differences between process models. Dependencies of an activity a indicates a set of activities that has an edge to a or a path to a without involving other activities (only gateways).

Definition 6 (Dependency (Set)). Given a business process graph g and an activity a of g . Let F consist of features containing a . The dependency set of a is defined as $D = \bigcup_{f \in F} \{a' | (a', a) \in E_f \vee (\exists gw_1, gw_2, \dots, gw_k \in GW_f, \{(a', gw_1), (gw_1, gw_2), \dots, (gw_{k-1}, gw_k), (gw_k, a)\} \subseteq E_f)\}$.

For example, the dependency set of activity d in “Graph a” in Fig. 2 is $\{a, b, c\}$.

Some feature can further be decomposed into features, e.g., feature $F1.1$ contains three subgraphs that are join features with two branches. However, we do not need to reconsider these subgraphs when we detect differences. Therefore, only local maximal features, as defined in Definition 7, need to be considered, e.g., feature $F1.1$.

Definition 7 (Local Maximal Feature). Let $g = (A, GW, E, \lambda)$ be a process graph. A subgraph of g $f = (A_1, GW_1, E_1, \lambda)$ is a feature. Feature f is a local maximal feature if and only if there exists no subgraph f' in g , such that f' is a feature of g and f is a subgraph of f' .

3 Detecting Differences with Features

This section presents how to detect differences between features. Before we define control-flow differences, two types of activity differences are presented, because they are the basis of comparing control-flows. These differences are the skipped node and the interchanged node, which is determined by whether a graph node can find a matching node in the other graph. Two nodes in two business process graphs with identical labels are considered as identical nodes. There are also other difference patterns between nodes [5]. For example, a graph node can be refined by a fragment (containing a group of nodes) in another graph. However, our focus is the control-flow aspect, and for node matching we only consider one node in a graph at a time to make it simple. There are researches focusing on matching two groups of nodes [18].

We analyze nodes based on their label similarities for illustration purposes. Label similarity can be measured in a number of different ways [7].

Definition 8 (Label Similarity, Node Matching). Let $G = (A, GW, E, \lambda)$ be a business process graph and $n, m \in A$ be two nodes and let $|x|$ represent the number of characters in a label x . The string edit distance of the labels $\lambda(n)$ and $\lambda(m)$ of the nodes, denoted $ed(\lambda(n), \lambda(m))$ is the minimal number of atomic string operations needed to transform $\lambda(n)$ into $\lambda(m)$ or vice versa. The atomic string operations are: inserting a character, deleting a character or substituting a character for another. The label feature similarity of $\lambda(n)$ and $\lambda(m)$, denoted $lsim(n, m)$ is:

$$lsim(n, m) = 1.0 - \frac{ed(\lambda(n), \lambda(m))}{\max(|\lambda(n)|, |\lambda(m)|)}$$

Let $lcutoff$ be a cutoff value for node matching, which can be assigned as desired. Given two nodes n and m , they are matched, denoted as $MATCH(n, m)$, if and only if their label similarity is no less than $lcutoff$, i.e., $lsim(n, m) \geq lcutoff$.

Given two business process graphs, a node is a skipped node if and only if a node in one business process graph does not match with any node in the other business process graph; while a node is an interchanged activity if and only if a node in one business process graph matches with a node in the other business process graph with a different label.

Definition 9 (Skipped Activity, Interchanged Activities). Let $g_1 = (A_1, GW_1, E_1, \lambda)$ and $g_2 = (A_2, GW_2, E_2, \lambda)$ be two business process graphs. Given an activity node $a_1 \in A_1$, a_1 is a skipped activity, if and only if there exists no activity node $a_2 \in A_2$ matches with a_1 , i.e., $\nexists a_2 \in A_2, \text{MATCH}(a_1, a_2)$; a_1 and a_2 are interchanged activities, if and only if there exists an activity node $a_2 \in A_2$ labeled differently matches with a_1 , i.e., $\exists a_2 \in A_2, \text{MATCH}(a_1, a_2) \wedge \lambda(a_1) \neq \lambda(a_2)$.

Fig. 4 shows an example for each difference considered in this paper. In the following content, each difference and how to detect it with features is explained.

Difference *different-conditions* indicates that a set of activities generate different traces in two similar business process models; while Difference *additional-conditions* indicates that a set of activities generate more traces in one business process model than the other. Definition 10 presents how to detect these two types of differences in the context of features. Given two features, there is an one-to-one mapping between two activity sets. If the trace sets of these two features are different, they are detected as *different-conditions*; if one trace set contains the other, they are detected as *additional-conditions*. For example, the first pair of features in Fig. 4 is detected as *different-conditions*, because their trace sets are $\{\langle a, b, c \rangle, \langle b, a, c \rangle\}$ and $\{\langle a, c \rangle, \langle b, c \rangle\}$ respectively; the second pair of features in Fig. 4 is detected as *additional-conditions*, the trace set $\{\langle a, b, c \rangle, \langle b, a, c \rangle\}$ is a subset of the other trace set $\{\langle a, b, c \rangle, \langle b, a, c \rangle, \langle a, c \rangle, \langle b, c \rangle\}$. Note that *additional-conditions* is a special case of *different-conditions*.

Definition 10 (Different/Additional Conditions). Given two features $f_1 = (A_1, GW_1, E_1, \lambda)$ and $f_2 = (A_2, GW_2, E_2, \lambda)$, T_1 and T_2 are their feature trace sets respectively. Suppose $|A_1| = |A_2|$ and there is a bijection $M : A_1 \rightarrow A_2$ for the activity sets of two features. f_1 and f_2 are detected as

- *different-conditions* if and only if $T_1 \neq T_2$;
- *additional conditions* if and only if $T_1 \subset T_2 \vee T_2 \subset T_1$.

In a process model, dependencies of a given activity indicates a set of activities that can enable the occurrence of the given activity. Difference *different-dependencies* indicates that in two business process models, an activity (a pair of interchanged activities) depends on two different sets of activities; difference *additional-dependencies* indicates that in two business process models, an activity (a pair of interchanged activities) depends on additional activities in one model besides activities the activity depend on in the other model; difference *disjoint-dependencies*¹ indicates that two business process models, an activity

¹ This type of difference is named as *different moment* in [5].

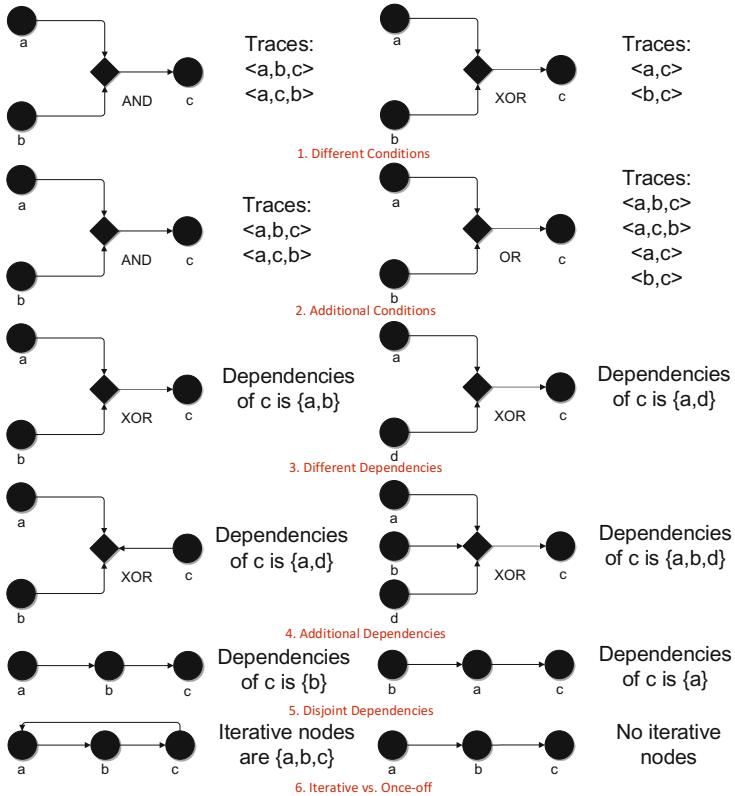


Fig. 4. Example of Behavioral Differences

(a pair of interchanged activities) depends on two disjoint sets of activities. Definition 11 presents how to detect these types of differences in the context of features. Given an activity and two business process models, a set of features containing the activity are gathered for each process model. Then, a set of activities that can appear right before but not after the activity are selected by checking edges of these features for each process model. If the two activity sets are different, they are detected as *different-dependencies*; if one activity set contains the other, they are detected as *additional-dependencies*. For example, the third pair of features in Fig. 4 is detected as *different-dependencies* for *activity c*, because in these two models dependencies of *activity c* are $\{a, b\}$ and $\{a, d\}$ respectively; the fourth pair of features in Fig. 4 is detected as *additional-dependencies* for *activity c*, because in these two models dependencies of *activity c* are $\{a, b\}$ and $\{a, b, d\}$ respectively and $\{a, b\}$ is a subset of $\{a, b, d\}$; the fifth pair of features in Fig. 4 is detected as *disjoint-dependencies* for *activity c*, because in these two models dependencies of *activity c* are $\{b\}$ and $\{d\}$ respectively, which are disjoint sets. Note that *additional-dependencies* and *disjoint-dependencies* are special cases of *different-dependencies*.

Definition 11 (Different/Additional/Disjoint Dependencies). Given two business process graph g_1 and g_2 , activity a is in both graphs or has an interchanged activity in the other graph, D_1 and D_2 are its dependencies in g_1 and g_2 respectively as defined in Definition 6. It is detected as different-dependencies in g_1 and g_2 if and only if $D_1 \neq D_2$; detected as additional-dependencies if and only if $D_1 \subset D_2 \vee D_2 \subset D_1$; detected as disjoint-dependencies if and only if $D_1 \cap D_2 = \emptyset$.

The last type of difference is named *iterative-vs.-once-off*, indicating an activity executed repeatedly in one business process model while executed only one time in the other. Definition 12 presents how to detect these types of differences in the context of features. Given an activity, it is in a loop feature of one business process model, while it is not in any loop feature of the other business process model, e.g., the last pair of features in Fig. 4.

Definition 12 (Iterative vs. Once-off). Given an activity a , F_1 and F_2 are sets of all features from two business process models. Suppose that there is no “dead” activity in F_1 and F_2 , i.e., every activity can be executed during process executions of the original process models. Activity a is detected as iterative-vs.-once-off if and only if

- $\exists f_1 \in F_1$ (F_2) is a loop feature containing activity a ;
- $\nexists f_2 \in F_2$ (F_1) is a loop feature containing activity a .

Some differences are subtypes of others and a detailed overview of the relations between these differences is provided in [4]. To avoid reporting multiple differences for one pair of activities, an order is given among these differences. The order is reversed with respect to the examples in Fig.4: *iterative-vs.-once-off*, *disjoint-dependencies*, *additional-dependencies*, *different-dependencies*, *additional-conditions*, and *different-conditions*.

All differences that can be detected by trace equivalence are detected by the feature-based technique proposed in this paper. To detect *iterative-vs.-once-off*, the trace-equivalence technique [5] checks whether an activity is executed multiple times in some traces of one trace set of a process model but at most once in all traces of the other. An activity can be executed multiple times if and only if it is a part of a loop feature. Therefore, this type of difference can be detected by checking loop features as defined in Definition 12. To detect *disjoint-dependencies*, *additional-dependencies*, and *different-dependencies*, the trace-equivalence technique computes the dependency set of an activity by collecting all activities that appear right before the activity in all traces and then check which activities of them can enable the activity. There are two cases that two activities appear next to each other in traces: first, these two activities are connected directly or through some gateways; second, they are in parallel branches. In our technique, the first case is covered as defined in Definition 11, while the second case two activities can be executed in different order and one cannot enable the other, therefore it does not satisfy the conditions of dependency. To detect *additional-conditions* and *different-conditions*, the trace-equivalence technique compares

trace sets of two process models while our technique compares traces sets of two features. Features are fragments of process models, therefore if trace sets of the pair of process models are different, at least trace sets of one pair of features are different, vise versa. The difference between the trace-equivalence technique and our technique for detecting *additional-conditions* and *different-conditions* is that the trace-equivalence technique can at most detect one difference of these two types for a pair of process models; while our technique can detect one for each pair of features of the process models. That is differences detected by our technique is more precisely related to a pair of small fragments instead of the whole models in the trace-equivalence technique.

4 Detecting Differences between Process Graphs

This section presents the steps to detect differences between business process graphs. There are totally three steps: feature splitting, activity matching, difference detecting.

The first step is to get all local maximal features, as described in Algorithm 1. A business process graph is traversed to detect and record all activities in loops. For each gateway, split, join, and complex features are selected and then sequence features are selected. A mapping from activity node to feature is used to quickly retrieve a feature. If an activity node a that has non-empty dependency set in a feature f , (a, f) is put in the mapping. Note that an activity node has at most one corresponding feature in which the activity has non-empty dependency set. The complexity of this step is $O(n^2)$. Take *graph a* in Fig. 2 as an example. During the traversals, no loop features in these graphs are found. Then, *graph a* is decomposed into two join features, $F1.1$, $F1.2$ and the mapping is $\{(d, F1.1), (f, F1.2)\}$. After selecting join features, no nodes are left in the graph, therefore, no sequence features are found in the graph.

The second step is to find a mapping between two activity sets by analyzing their labels. The greedy algorithm is applied to find such a mapping, as described in the top part of Algorithm 2. The time complexity of this step is $O(n^3)$, where n is the number of activity nodes in a process graph. Take the pair of graphs in Fig. 2 as an example. *Activity a*, d , e and f are identical activities between these two graphs; *activity b* and b' (“*Inbound Delivery Entered*” and “*Inbound Delivery Created*”) are interchanged activities; *activity c* is a skipped activity.

Lastly, differences are detected as described in Algorithm 3. For each pair of identical activities in the two process graphs, differences are checked one by one in the order of *iterative-vs.-once-off*, *disjoint-dependencies*, *additional-dependencies*, and *different-dependencies*. It stops when a difference is detected or these four differences do not apply. Then, for the pair of activities that is not detected as any above differences, trace sets are computed for the corresponding features in the mapping F for the activities. These features and traces

Algorithm 1. Local Maximal Features

```

input : a process graph:  $g = ((A, GW), E, \lambda)$ ;
output: a mapping from activity node to feature,  $F$ ; activity node set  $LOOP$ 
1 begin
2    $LOOP \leftarrow \emptyset; F \leftarrow \emptyset;$ 
3 foreach  $n \in A$  do
4    $\left[ \begin{array}{l} \text{if there exists a path from } n \text{ to } n \text{ then } LOOP \leftarrow LOOP \cup \{n\} \end{array} \right]$ 
5 while  $GW \neq \emptyset$  do
6   Randomly select  $gw \in GW$ ;
7   Let  $f$  be the local maximal (complex) join/split feature in  $g$  containing
8      $gw$  (Definition 4, 5 and 7);
9   foreach  $a \in A_f \wedge \exists(x, a) \in E_f$  do  $F \leftarrow F \cup \{(a, f)\}$ 
10   $A \leftarrow A - \{a \in A_f \mid \nexists(a, x) \in (E - E_f) \wedge (\exists(x, a) \in (E - E_f))\};$ 
11   $GW \leftarrow GW - GW_f; E \leftarrow E - E_f;$ 
12 while  $A \neq \emptyset$  do
13   Randomly select  $a \in A$ ;
14   Let  $f$  be a local maximal sequence feature in  $g$  (Definition 4 and 7);
15   foreach  $a \in A_f \wedge \exists(x, a) \in E_f$  do  $F \leftarrow F \cup \{(a, f)\}$   $A \leftarrow A - A_f$ ;
16    $GW \leftarrow GW - GW_f; E \leftarrow E - E_f;$ 
17 return  $(F, LOOP);$ 

```

Algorithm 2. Node Mapping

input : two sets of nodes: A_1, A_2
output: a mapping between two input sets of nodes: M

```

1 begin
2    $openpairs \leftarrow A_1 \times A_2; M \leftarrow \emptyset;$ 
3 while  $\exists(n, m) \in openpairs \wedge lsim(n, m) \geq lcutoff \wedge \nexists(x, y) \in$ 
4      $openpairs, lsim(x, y) > lsim(n, m)$  do
5      $M \leftarrow M \cup \{(n, m)\};$ 
6      $openpairs \leftarrow \{(x, y) \in openpairs \mid x \neq n, y \neq m\};$ 
7 return  $M;$ 

```

are compared to detect difference *additional-conditions*, and *different-conditions*. The complexity of this step is also $O(n^2)$. For example, in Fig. 2, *activity d* is detected as *additional-dependencies*, because the dependency set in *graph b* is a subset of that in *graph b*, i.e., $\{a, b\} \subset \{a, b, c\}$. *Activity f* is detected as *different-conditions*, because the dependency set in both graphs are $\{d, e\}$, while the trace set of *feature F1.2* is $\{\langle d, f \rangle, \langle e, f \rangle\}$ and the trace set of *feature F2.2* is $\{\langle d, e, f \rangle, \langle e, d, f \rangle\}$.

Algorithm 3. Difference Detection

```

input : two process graphs:  $g_1, g_2$ 

1 begin
2    $M = \text{getNodeMapping}(A_1, A_2);$ 
3    $(F_1, LOOP_1) = \text{getFeature}(g_1); (F_2, LOOP_2) = \text{getFeature}(g_2)$ 
      (Algorithm 1);
4 foreach  $(a_1, a_2) \in M$  do
5   if  $(a_1 \in LOOP_1 \wedge a_1 \notin LOOP_2) \vee (a_1 \notin LOOP_1 \wedge a_1 \in LOOP_2)$  then
6      $\text{Difference} \Leftarrow \text{Difference} \cup ((a_1, a_2), \text{"Loop2Onceoff"})$ 
7 else
8   Let  $f_1, f_2$  satisfy that  $(a_1, f_1) \in F_1 \wedge (a_2, f_2) \in F_2$ ;
9   Let  $D_1, D_2$  be the dependency set of  $a_1, a_2$  in  $f_1, f_2$  (Definition 11) ;
10  if  $D_1 \cap D_2 = \emptyset$  then
11     $\text{Difference} \Leftarrow \text{Difference} \cup ((f_1, f_2), \text{"DisjointDependencies"})$ 
12 else if  $D_1 \subset D_2 \vee D_2 \subset D_1$  then
13     $\text{Difference} \Leftarrow \text{Difference} \cup ((f_1, f_2), \text{"AdditionalDependencies"})$ 
14 else if  $D_1 \neq D_2$  then
15     $\text{Difference} \Leftarrow \text{Difference} \cup ((f_1, f_2), \text{"DifferentDependencies"})$ 
16 else if  $D_1 = D_2$  then
17   Let  $T_1, T_2$  be the trace set of  $f_1, f_2$  (Definition 3) ;
18   if  $T_1 \subset T_2 \vee T_2 \subset T_1$  then
19      $\text{Difference} \Leftarrow \text{Difference} \cup ((f_1, f_2), \text{"AdditionalConditions"})$ 
20 else if  $T_1 \neq T_2$  then
21      $\text{Difference} \Leftarrow \text{Difference} \cup ((f_1, f_2), \text{"DifferentConditions"})$ 

```

5 Evaluation

This section presents the experiment we run to evaluate the technique using a real-life process model collection. It first explains the setup of the evaluation and second the results. Also statistics are provided on the differences detected from more than 140,000 pairs of similar process models.

Business process models that are used in the experiments are derived from the SAP reference model. This is a collection of 604 business process models (described as EPCs) capturing business processes supported by the SAP enterprise system. On average the process models contain 21.6 nodes with a minimum of 3 and a maximum 130 nodes. The average size of node labels is 3.8 words. All the experiments are run on a laptop with the Intel Core2 Duo processor T7500 CPU (2.2GHz, 800MHz FSB, 4MB L2 cache), 4 GB DDR2 memory, the Windows 7 operating system and the SUN Java Virtual Machine version 1.6.

Similarities of any pair among 604 business process models are computed [22]. Pairs with similarity higher than or equal to 0.5 are selected for the experiment, because more differences are expected if two models are somehow similar with

variations. There are more than 140,000 pairs of process models in total. We run the experiment five times with our difference detection technique and the average execution time is 1652.2s for all pairs. On average it takes 0.012s to detect differences for each pair and in the worst case it takes only 1.1s.

Table 1 shows the statistics of differences detected in our experiments. The columns show the cutoff value that used to determine whether two labels are matching. Five different values are used, from 0.5 to 0.9. The rows show different types of differences considered in this paper. Except for *skipped-activity*, *disjoint-dependencies* is the most frequent difference, which indicates an activity has completely different activities as its enabler two models. Also, in many cases, an activity is in a loop in one model, but is not in the other. Difference *additional-conditions* and *different-conditions* are barely found. This is because they are only considered if other differences do not apply. As expected, when the cutoff value for label matching decreases, more activities are matched and more differences are detected except for *skipped-activity*.

Table 1. Statistics of differences between process models

Difference	lsim ≥ 0.9	lsim ≥ 0.8	lsim ≥ 0.7	lsim ≥ 0.6	lsim ≥ 0.5
Skipped activity	4814309	4814104	4804352	4776779	4691821
Interchanged activities	97	118	156	189	240
Iterative vs. once-off	536	558	587	812	1732
Disjoint Dependencies	1227	1407	2359	5223	14117
Additional Dependencies	235	243	269	312	356
Different Dependencies	147	202	210	236	228
Additional Conditions	2	2	2	8	10
Different Conditions	12	9	8	14	14

From the experiments we conclude that our technique can detect behavioral differences efficiently, at least four orders of magnitude faster than existing techniques [5]. It reduces the computations from hours to about one second in worst cases. The differences are valuable for analyzing variations between similar business models in an organization. The differences can be further applied to editing one process model to another or abstracting a template for a set of similar process models.

6 Related Work

The work presented in this paper is related to business process difference detection techniques, business process alignment techniques, and business process retrieval techniques.

Existing researches about difference detection follow into two branches: detecting semantic or syntactic differences between process models. Dijkman summarizes semantic differences [3] and proposes a technique based on trace equivalence to detect these differences [5]. In this paper, we propose a technique

based on process fragments to enhance the efficiency. On the other hand, Li et al. [15], Liu et al. [16] and Küster et al. [13] focus on structures of business process models and proposes techniques to identify syntactic differences between these models. Features are also used to detect differences in [16]. However, [16] focuses on structural differences, while this paper focuses on behavioral differences. Furthermore, flexible feature matching based on structures are used to detect structural differences in [16], while dependencies and traces of features are used to detect behavioral differences in this paper.

Process alignment is complementary to process difference detection. The former techniques align the matching parts and the latter techniques detect the mismatching parts between process models. Weidlich et al. [18,19] provide techniques to align models for the same process at different abstraction level based on the semantics of these models. Dijkman et al. [6] propose to align process models based on the syntax of models.

Another relevant research topic is process retrieval. Process difference detection and alignment techniques focus on comparing a pair of process models; while process retrieval techniques focus on comparing one given query model with a collection of process models. There are basically two branches of process retrieval: process similarity search [7,8,12,22] and process querying [1,2,10,11,23]. In previous work, features are also used to build an index for process retrieval [20,23]. However, in these work features of the same structure are used to filter models in a collection, while in this paper features of different behaviors are considered to detect behavioral differences between two process models.

7 Conclusion

This paper presents a technique for detecting behavioral differences between business process models. Compared with the previous technique [5], the technique in this paper improves the efficiency of the detection by at least four orders of magnitudes. Especially in the worst cases, the execution time decreases from hours to about one second. The technique works by decomposing process models into small fragments, i.e., sequences, splits, joins, and loops, and then analyzing execution semantics based on these fragments.

There are some possible improvements on the technique described in this paper, which are left for future work. First, we assume that all activities of given process models can be executed, which may also not always be the case. Such activities may affect the accuracy of our technique. However, there are techniques [17] that can verify process models. These techniques can be integrated with our technique to mark activities that cannot be executed or be used to filter unsound process models. Second, the technique in this paper mainly focuses on activities and control-flows. However, process models often contain more information, e.g., resources and data used, which are also useful for detecting differences between models. Third, loops are identified by analyzing paths on the process models in this paper, which cannot distinguish whether a model contains a simple loop or nesting of loops. Last, the difference detection technique in this

paper can be further applied to editing models based on features or abstracting a template model for a set of similar process models.

Acknowledgement. The research is supported by the MOE-CMCC research foundation project No. MCM20123011, the research fund of Capital University of Economics and Business, project #00791462722336 and #00791465730165, and the special fund for innovation of Shandong, China project No.2013CXC30001.

References

1. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.T.: Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
2. Awad, A.: BPMN-Q: A language to query business processes. In: Proceedings of EMISA 2007, Nanjing, China, pp. 115–128 (2007)
3. Dijkman, R.: A classification of differences between similar Business Processes. In: EDOC 2007, pp. 37–37 (2007)
4. Dijkman, R.: Feedback on Differences between Business Processes. BETA Working Paper WP-234, Eindhoven University of Technology, The Netherlands (2007)
5. Dijkman, R.: Diagnosing differences between business process models. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 261–277. Springer, Heidelberg (2008)
6. Dijkman, R., Dumas, M., Garcia-Banuelos, L., et al.: Aligning business process models. In: IEEE International Enterprise Distributed Object Computing Conference, EDOC 2009, pp. 45–53. IEEE (2009)
7. Dijkman, R., Dumas, M., Van Dongen, B., et al.: Similarity of business process models: Metrics and evaluation. Information Systems 36(2), 498–516 (2011)
8. Dijkman, R.M., van Dongen, B.F., Dumas, M., et al.: A Short Survey on Process Model Similarity. In: Seminal Contributions to Information Systems Engineering - 25 Years of CAiSE, pp. 421–427 (2013)
9. Gao, X., Chen, Y., Ding, Z., et al.: Process Model Fragmentization, Clustering and Merging: An Empirical Study. In: BPM workshops: The 4th International Workshop on Process Model Collection: Management and Reuse, Beijing, China (2013)
10. ter Hofstede, A.H.M., Ouyang, C., La Rosa, M., et al.: APQL: A process-model query language. In: Asia Pacific Business Process Management, pp. 23–38 (2013)
11. Jin, T., Wang, J., Wu, N., La Rosa, M., ter Hofstede, A.H.M.: Efficient and Accurate Retrieval of Business Process Models through Indexing. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 402–409. Springer, Heidelberg (2010)
12. Kunze, M., Weidlich, M., Weske, M.: Behavioral Similarity – A Proper Metric. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 166–181. Springer, Heidelberg (2011)
13. 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)

14. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., Garcia-Banuelos, L.: APROMORE: an advanced process model repository. *Expert Systems with Applications* 38(6), 7029–7040 (2011)
15. Li, C., Reichert, M.U., Wombacher, A.: On Measuring Process Model Similarity based on High-level Change Operations. In: Proceedings of the 27th International Conference on Conceptual Modeling, Barcelona, Spain, pp. 248–264 (2008)
16. Liu, K., Yan, Z., Wang, Y., Wen, L., Wang, J.: Efficient Syntactic Process Difference Detection using Flexible Feature Matching. In: The 2nd Asia Pacific Conference on Business Process Management (to appear, 2014)
17. M.J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. LNBP, vol. 6. Springer, Heidelberg (2008)
18. Weidlich, M., Weske, M., Mendling, J.: Change propagation in process models using behavioral profiles. In: SCC 2009, pp. 33–40. IEEE (2009)
19. Weidlich, M., Dijkman, R.M., Mendling, J.: The ICoP Framework: Identification of Correspondences between Process Models. In: Proceedings of the 22th CAiSE, Hammamet, Tunesia (2010)
20. Yan, Z., Dijkman, R., Grefen, P.: Fast business process similarity search with feature-based similarity estimation. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 60–77. Springer, Heidelberg (2010)
21. Yan, Z., Dijkman, R.M., Grefen, P.W.P.J.: Business Process Model Repositories - Framework and Survey. *Information and Software Technology* 54(4), 380–395 (2012)
22. Yan, Z., Dijkman, R.M., Grefen, P.W.P.J.: Fast Business Process Similarity Search. *Distributed and Parallal Databases* 30(2), 105–144 (2012)
23. Yan, Z., Dijkman, R., Grefen, P.: FNet: An index for advanced business process querying. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 246–261. Springer, Heidelberg (2012)

Compliance Checking of Data-Aware and Resource-Aware Compliance Requirements

Elham Ramezani Taghiabadi, Vladimir Gromov, Dirk Fahland,
and Wil M.P. van der Aalst

Eindhoven University of Technology, The Netherlands
`{e.ramezani,v.gromov,d.fahland,w.m.p.v.d.aalst}@tue.nl`

Abstract. Compliance checking is gaining importance as today's organizations need to show that their business practices are in accordance with predefined (legal) requirements. Current compliance checking techniques are mostly focused on checking the control-flow perspective of business processes. This paper presents an approach for checking the compliance of observed process executions taking into account data, resources, and control-flow. Unlike the majority of conformance checking approaches we do not restrict the focus to the ordering of activities (i.e., control-flow). We show a collection of typical data and resource-aware compliance rules together with some domain specific rules. Moreover providing diagnostics and insight about the deviations is often neglected in current compliance checking techniques. We use control-flow and data-flow alignment to check compliance of processes and combine diagnostics obtained from both techniques to show deviations from prescribed behavior. Furthermore we also indicate the severity of observed deviations. This approach integrates with two existing approaches for control-flow and temporal compliance checking, allowing for multi-perspective diagnostic information in case of compliance violations. We have implemented our techniques and show their feasibility by checking compliance of synthetic and real life event logs with resource and data-aware compliance rules.

Keywords: compliance checking, auditing, data-aware and resource-aware compliance requirements, conformance checking.

1 Introduction

Organizations need to comply with an increasing set of laws, regulations, and service level agreements set by both internal and external stakeholders. Major corporate and accounting scandals including those affecting Enron, Tyco, Adelphia, Peregrine, and WorldCom have fueled the interest in more rigorous auditing practices. Legislation, such as the Sarbanes-Oxley (SOX) Act of 2002 and the Basel II Accord of 2004, was enacted as a reaction to such scandals. Failing to comply may be costly, therefore organizations need to continuously check whether processes are executed within a given set of boundaries. Moreover organizations seek for a better control of their processes to streamline their business operation and prevent fraud, malpractices, risks, and inefficiencies.

There are two basic types of compliance checking: (1) *forward compliance checking* aims to design and implement processes where compliant behavior is enforced, and (2) *backward compliance checking* aims to detect and localize non-compliant behavior. This paper focuses on backward compliance checking based on event data.

Current backward compliance checking techniques focus on verifying aspects related to control flow while checking compliance requirements addressing other fundamental

aspects of a process including data handling, and resources are as important. The compliance requirements considered in this paper take into account data, resources, control-flow, and their interplay. We have collected several compliance requirements found in literature; we classify these requirements using two main categories and propose two generic techniques for checking these rules. Our approach seamlessly integrates with control-flow compliance checking. More important, the technique provides detailed diagnostic information in case of non-compliant behavior; it shows for each process instance *which attribute(s)* (resource or data) in *which event* violated a requirement and *what* changes would have been needed to make the behavior compliant. Our data and resource-aware compliance checking techniques leverage a recent *data-aware* alignment technique [11] that allows to check conformance of a log with respect to a *data-aware Petri net*. We adapt and improve this technique for our purpose. Moreover, our collection of data and resource-aware compliance rules, address the problem of compliance rule elicitation from informal description of compliance requirements.

The remainder of this paper is organized as follows. We recall some basic definitions for control-flow and data-flow alignments in Sect. 2. In Sec. 3, we give an overview on how a compliance requirement may impact different perspectives of a business process and show different types of data and resource-aware compliance rules. Section 4 introduces the problem of data and resource-aware compliance checking and discusses our solution through a running example. In Sect. 5 the implementation of the approach in ProM is showcased. Experimental results are presented and discussed in Sect. 6. We discuss related work in Sect. 7, and Sect. 8 concludes this paper.

2 Preliminaries

This section recalls basic notions for control-flow alignment [2,5] and data-flow alignment [12] on which we build for data and resource-aware compliance checking. Alignments relate recorded executions of a process to a formal specification which describes the boundaries of a compliant process.

Logs. Executions of business processes are recorded in terms of *event logs*. An event log L is a multiset of traces. A log trace $\sigma^L = \langle e_1, \dots, e_n \rangle \in L$ describes a particular process execution as a sequence of events. An event e refers to the execution of an activity and is a tuple of pairs of distinct attributes and their corresponding values; $e = ((attr_1^e, val_1^e), \dots, (attr_k^e, val_k^e))$. Each attribute and its corresponding value provides information on the executed activity. For instance in the event $e = ((name, approve\ loan\ request), (resource, John), (amount, 10000))$, the pair $(name, approve\ loan\ request)$ refers to the name of the respective activity executed, $(resource, John)$ indicates the resource approved the request, and $(amount, 10000)$ records the request amount.

Let $E = \{e_1, \dots, e_n\}$ be the set of all events in log L . $ATTR^e = \{attr_1^e, \dots, attr_k^e\}$ is the set of all k attributes of an event e ; and $VAL^e = \{val_1^e, \dots, val_k^e\}$ is the set of all values for attributes of an event e . As a shorthand, we write $l^e(attr_i^e) = val_i^e$ to denote that event e has value val_i^e for attribute $attr_i^e$, $1 \leq i \leq k$. We define $ATTR^L$ as the set of all event attributes in the log; $ATTR^L = \cup_{e \in E} ATTR^e$.

Specified Behaviors. A specification describes what is considered compliant and what is not. In essence, each specification describes a set S of *compliant traces*. Every compliant

trace $\sigma^S = \langle a_1, \dots, a_n \rangle \in S$ is a sequence of activities which are allowed to be executed. Activities may have attributes with corresponding admissible values that describe how an activity must be executed in a compliant trace. An activity a is a tuple of pairs of distinct attributes and their corresponding set of admissible values; $a = ((attr_1^a, Val_1^a), \dots, (attr_k^a, Val_k^a))$ where each Val_i^a is a set of admissible values that attribute $attr_i^a$ is allowed to take.

For instance for the activity $a = ((name, \{ approve \ loan \ request \}), (resource, \{ John, Elham, Luis \}), (amount, \{ 500, \dots, 1500 \}))$, the pair $(name, \{ approve \ loan \ request \})$ shows the name of the activity allowed to be executed, $(resource, \{ John, Elham, Luis \})$ indicates the resources who are allowed to approve a loan request, $(amount, \{ 500, \dots, 1500 \})$ specifies the admissible amount for a loan.

As a shorthand, we write $l^a(attr_i^a) = Val_i^a$ to denote that activity a can take values Val_i^a for attribute $attr_i^a$, $1 \leq i \leq k$. The set $ATTR^a$ of all attributes of activity a , and $ATTR^S$ of specification S are defined similar to attributes of events and logs.

Given an activity a , l^a assigns a set of admissible values to an attribute of activity a ; $l^a(attr_i^a) = Val_i^a$ for $1 \leq i \leq k$. We define $ATTR^a$ as the set of all k attributes of an activity a ; $ATTR^a = \{ attr_1^a, \dots, attr_k^a \}$. $ATTR^S$ is the set of all attributes of all the activities in the specification; $ATTR^S = \cup_{a \in A} ATTR^a$, and A is the set of all activities in specification S ; $A = \{ a_1, \dots, a_n \}$.

Relating Events in a Log to Activities in the Specification. As we already mentioned every event refers to execution of an activity. We will explain later in this paper how we use alignment techniques for checking compliance. There, we will pair events in the log and activities in the specification using the following notation.

For event log L with events E , and attributes $ATTR^L$, we call an attribute $type^L \in ATTR^L$ a *log type attribute* if $type^L \in \cap_{e \in E} ATTR^e$, i.e., an attribute present in all events in L . We assume that at least one $type^L$ exists for L . We define $TYPEVAL^L$ as the set of all values for $type^L$; $TYPEVAL^L = \{ l^e(type^L) | e \in E \}$.

Similarly for specification S with activities A and attributes $ATTR^S$, we call an attribute $type^S \in ATTR^S$ a *specification type attribute* if $type^S \in \cap_{a \in A} ATTR^a$, i.e., an attribute present in all activities of specification S . We assume that at least one $type^S$ exists for specification S and attribute $type^S$ has only a single value for each activity i.e., $l^a(type^S) = \{ label^a \}$. We define $TYPEVAL^S$ as the set of all admissible values for a specification attribute $type^S$; $TYPEVAL^S = \cup_{a \in A} l^a(type^S)$.

Note that a log and a specification may have multiple type attributes. Later on, we relate L to S by picking a specific log type attribute and a specific specification type attribute each, and mapping the values $TYPEVAL^L$ to the values $TYPEVAL^S$.

Data-Aware Petri Nets. The specification S representing all compliant traces, can be expressed in various ways. For instance as a Petri net [2] or in terms of declarative constraints [12]. Typically, the specification S is very large, being the semantic notion of all compliant traces. In this paper, we use Petri nets to specify S in a concise form. Fig. 1 illustrates a variant of Petri nets called *data-aware* Petri net. Data-aware Petri nets extend classical Petri nets with data. They describe which activities are allowed to be executed and in which sequence. In addition, a data-aware Petri net describes how every activity is allowed to be executed with respect to its attributes and their values. N^S —the data-aware Petri net shown in Fig. 1—describes a simplified version of a loan application procedure.

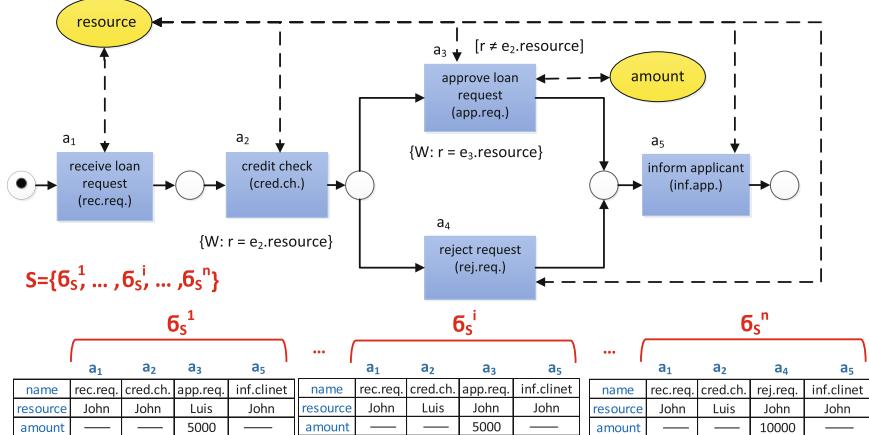


Fig. 1. N^S , a data-aware Petri net example, and some compliant trace examples from specification S

The process starts with activity a_1 which is represented with the transition *receive loan request* and continues with a_2 represented by *check credit* of the applicant requesting the loan. Afterwards the decision for *approving* or *rejecting* the request should be taken either by executing activity a_3 or a_4 , and finally the applicant will be *informed* about the decision by executing activity a_5 . Each activity has an attribute *name* that we will use later as a specification attribute $type^S$ for mapping (please see Sect.2). The admissible value of this attribute is written inside the transition representing each activity.

Moreover N^S depicts other admissible attributes as the yellow colored eclipses including *resource*, and *amount*. Some activity attributes may be accessible by all the activities e.g., *resource* or may be accessible by specific activities e.g., *amount*. This is specified by the *dashed line* connecting attributes to activities. A specification describes how an activity must be executed in order to be compliant; hence in a data-aware Petri net activities may have additional annotations. For instance in N^S , activity a_3 represented by the labeled transition *approve loan request* is guarded. The guard restricts the relation between activities of a_2 , and a_3 with the help of variable r . The value of the attribute *resource* for the recent event produced by the execution of a_2 is stored in variable r . This is expressed by the *write statement* $\{r = e_2.resource\}$ annotated at a_2 . The Execution of activity a_3 updates the value of r . This is expressed by the *write statement* $\{r = e_3.resource\}$ annotated at a_3 . The guard on activity a_3 $[r \neq e_2.resource]$ specifies that the *resource* executed a_3 must be different with the *resource* executed a_2 . That is, the new value of r must be different from its previous value which was written by the execution of a_2 . Please note that this guard describes *four-eye principle*. The firing sequences of the Petri net N^S is the set S given in Fig. 1(bottom).

Aligning Observed Behavior to Specified Behavior.

An observed trace in a log may deviate from admissible behaviors; e.g., the non-compliant trace σ^L shown in Fig. 2. There are three events and three attributes in

	name	rec.req.	cred.ch.	app.req.
L	resource	John	Luis	Luis
	amount	—	—	10000

Fig. 2. Sample trace σ^L

this trace. We choose *name* as log type attribute $type^L$ (see Sect.2) and map the values of this attribute occurring in σ^L to transition labels in N^S when they have the same value. Please note that transition labels in N^S denote the admissible values of $type^S$. If we only consider the compliance of σ^L w.r.t. existence and correct sequence of events, we see that an event e with (*name:inform applicant*) is missing as the final event in the trace. To understand where σ^L deviates from specification S , we apply *control-flow alignment* [2] between σ^L and N^S as follows.

The idea is to find a compliant trace $\sigma^S \in S$ that is as similar as possible to σ^L ; differences between σ^S and σ^L then indicates deviations. We relate σ^L to any trace $\sigma^S \in S$ by pairing events in σ^L to activities in S where events and activities are of the same type with the same value. For our example $type^L = type^S = name$. The obtained alignment is shown in Fig. 3; top row corresponds to events in σ^L and bottom row refers to activities in σ^S . As is indicated in γ^c , an event with (*name:inform client*) is missing; this is denoted by \gg in the alignment indicating a move on model.

Let L be an event log and S be a specification. We pick a log type attribute $type^L$ (with values $TYPEVAL^L$ occurring in L) and a specification type attribute $type^S$ (with values $TYPEVAL^S$ occurring in S) as described above. Further, we pick a mapping $R : TYPEVAL^L$ to $TYPEVAL^S$ relating log type attributes to specification type attributes. A control-flow alignment move m^c (wrt. $type^L$, $type^S$, and R) is a pair $(x, y) \in (E \cup \{\gg\}) \times (A \cup \{\gg\}) \setminus \{(\gg, \gg)\}$ where:

- (x, \gg) is a *move on log*.
- (\gg, y) is a *move on specification S*.
- (x, y) is a *synchronous move* if $x, y \neq \gg$ and $R(l^x(type^L)) \in l^y(type^S)$.

A synchronous move (x, y) relates the event x to the activity y based on the log type attribute $type^L$ and the specification type attribute $type^S$; the type value of x has to map to the type value of y (via mapping R).

A control-flow alignment of a trace $\sigma^L \in L$ to S is a sequence $\gamma_c = \langle m_1^c, \dots, m_n^c \rangle$ of control-flow alignment moves such that ignoring \gg , the projection $x_1 \dots x_n|_E$ is the original trace σ^L , and the projection $y_1 \dots y_n|_A = \sigma^S \in S$ is described by the specification.

The control-flow conformance checking technique of [2,5,14] returns an *optimal* control-flow alignment s.t. no other alignment has fewer non-synchronous moves (move on log or move on specification only). This technique finds an optimal alignment using a cost-based approach: a cost function κ_c assigns each control-flow alignment move (x, y) a cost $\kappa_c(x, y)$ s.t. a synchronous control-flow alignment move has cost 0 and all other types of moves have cost > 0. Then an A^* -based search on the space of (all prefixes of) all alignments of σ^L to S is guaranteed to return an optimal alignment for σ^L and S . In such an optimal alignment, a move on log (e, \gg) indicates that the trace σ^L had an event

	name	rec.req.	cred.ch.	app.req.	
L	resource	John	Luis	Luis	\gg
	amount	—	—	10000	
S	name	rec.req.	cred.ch.	app.req.	inf.client

Fig. 3. A control-flow alignment γ_c of trace σ^L and specification S with $type^L = type^S = name$ and mapping: $R(val) = val$

e that was not supposed to happen according to the specification S whereas a move on specification (\gg, a) indicates that σ^L was missing an event that was expected according to S . As the alignment preserves the position relative to the trace σ^L , we can locate exactly where σ^L had an event too much or missed an event compared to S .

A data-aware alignment [11] extends a control-flow alignment by also comparing every event $e \in \sigma^L$ with all its attributes and their values to its corresponding activity $a \in \sigma^S$ with its admissible attributes and their values. If an event e was not executed in compliance with its corresponding activity, the data-aware alignment technique can identify the deviation and extent of the deviation.

Given event log L and specification S , a control-flow alignment $\gamma_c = \langle m_1, \dots, m_n \rangle$ of a log trace $\sigma^L \in L$ to S extends to a data-aware alignment $\gamma_d = \langle m_1, \dots, m_n \rangle$ by considering all attributes as follows. Log move and model move are defined as before. A synchronous move (e, a) is *correct* iff for each attribute $attr_i \in ATTR^e$ holds: $attr_i \in ATTR^a$ and $l^e(attr_i) \in l^a(attr_i)$, i.e., each data attribute of the event has a value admitted by the activity. Otherwise, the synchronous move is called *incorrect*. Figure 4 shows an example of a data-aware alignment.

Activity a_3 named *approve loan request* is executed by a *resource* which is not allowed based on the guard at a_3 in N^S . The resource (*Luis*) executed *approve loan request* should have been different from the resource executed *credit check*. In addition to finding deviations, γ_d indicates the correct value for the violating attribute. In our example, from the resources allowed to execute activity *approve loan request*, *John* is suggested as the correct value.

Similar to the cost-based approach applied in control-flow alignment, a cost function κ_d assigns each data-alignment move a cost $\kappa_d(x, y)$ s.t, a correct data-alignment move has cost 0 and an incorrect move has cost > 0. In the data-aware alignment technique of [11], an ILP solver finds among all synchronous control-flow alignment moves, values for attributes of S such that the total cost of deviations for an alignment including *move on log*, *move on model*, and *incorrect move* is minimized. We apply control-flow and data-aware alignments for data and resource-aware compliance checking.

3 Compliance Requirements

Compliance requirements prescribe how internal or cross-organizational business processes have to be designed or executed. They originate from legislations and restrict one or several perspectives of a process (control flow, data flow, process time or organizational aspects). Restrictions can be imposed for individual cases or groups of cases, they can prescribe properties of process executions or process design [14]. These different aspects of compliance give rise to the framework shown in Fig. 5. A complex compliance requirement covering several perspectives of a process can be decomposed into smaller compliance rules, each covering a single aspect along the dimensions of this framework.

	name	rec.req.	cred.ch.	app.req.	
L	resource	John	Luis	Luis	>>
	amount	—	—	10000	

	name	rec.req.	cred.ch.	app.req.	inf.client
S	resource	John	Luis	John	John
	amount	—	—	10000	—

Fig. 4. The data-aware alignment γ_d of trace σ^L and σ^S

For example, a compliance requirement might state: “Before approving a loan request, the bank must check the credit history of the applicant. A request can be approved if it passes the evaluation. The approval and credit checking must be done by different agents. Regardless of rejection or approval of the request, applicant must be informed about the bank’s decision within a week after submitting the request”.

This requirement can be divided into different compliance rules: *i) control flow*: ‘Loan request approval or rejection must be preceded by credit check’, *ii) process data*: ‘A loan may be approved only if it passes the evaluation’, *iii) process resource*: ‘Request approval and check credit must be done with different resources’, and *iv) process time*: ‘Every application for requesting a loan must be processed within one week from the date of application submission’; each rule taking only one perspective into account. Our earlier compliance checking techniques of [14,20] are able to check control-flow and temporal compliance rules, but do not provide a notion of *data* or *resource*. In the following, we present a new approach for checking data and resource-aware compliance rules that provides diagnostic information about deviations. Before, we will explain the data and resource-aware rules supported by our approach.

3.1 Data-Aware and Resource-Aware Compliance Rules

We identified some works [22,4,18,7,21,8,19] discussing compliance rules that restrict process data and resource. Table 1 shows the collection of compliance rules taken from these sources and some more taken from practise e.g., medical guidelines. We found some typical restrictions on process data and resources such as four-eye principle (separation of duties), authorization level or three-way match and some domain specific compliance rules.

In addition to the classification presented in Table 1, all different types of data-aware and resource-aware constraints fall into two main categories: (1) constraints that enforce a restriction on data attributes, and (2) constraints that restrict activities when a certain data condition holds. For example a compliance rule such as the *four-eye principle* is of the first category. This rule specifies that two activities *A* and *B* must be executed by two different resources. The rule assumes that the underlying control-flow sequence is correct i.e., no matter in which order two activities *A* or *B* are executed, the restriction is on the corresponding data attribute (resource). In case *A* is executed first by resource *R*₁, the rule will urge that *B* must not be executed by *R*₁. Whereas, if *B* was executed first, the restriction would be on the resource executing activity *A*. In contrary with rules like *four-eyes principle*, a rule stating “Activity *B* must not be executed for *gold* customers” belongs to the second category. This rule restricts the execution of activity *B* when a certain value (*gold*) for data attribute *customer type* holds. That is, based on a certain data condition, the constraint restricts the control-flow i.e., restricting execution of activity *B*.

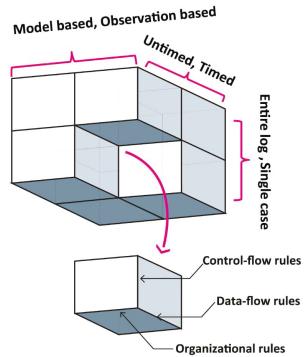


Fig. 5. Compliance Rule Framework [14]

In Sect. 2 we discussed an example of a rule from the first category. In the next section we will discuss examples of both categories in more detail while elaborating on the differences in corresponding checking technique employed.

4 Data-Aware and Resource-Aware Compliance Checking

This section presents our main contribution, an approach for checking data and resource-aware compliance on past executions recorded in event logs. We first introduce a motivating example which we use throughout this section to explain our techniques.

Rule Description	Example
Four-eye principle: A security principle that requires segregating the execution of critical tasks and associated privileges among multiple users.	The person requesting purchase of goods should not be the one who approves it. A purchase order approval requires two signatures.
Authorization (Access control): A security principle that limits execution of activities or accessing a data object to authorized individuals.	Only a financial manager can approve a loan.
Two (three)-way match :An accounting rule that requires the value of two different data objects to match.	All vendor invoices that are based on purchase orders should be matched with purchase order lines (two-way matching).
Activity T may/must (not) be executed if attribute X has the value v ; (X may be local to the activity T or may appear anywhere in a trace).	An account must not be opened in case risk is high. During ventilation, patient must receive “propofol” with dosage of (5mg).
Activity T_1 may/must (not) be executed if attribute X has value v at activity T_2 . (attribute X is local to activity T_2)	In case the respondent bank rating review is rejected during evaluation, an account must never be opened.
Activity T must not change value of attribute X .	Bank account data must not change during payment.
Value of attribute X must not change after activity T is executed.	All invoices must be archived and no changes must be made to the document.
Activity T_1 may occur only if the value of attribute X is increased/decreased by activity T_2 with d .	If gastric tube feeding cannot be increased by (1,20 kcal/ml), then use ‘Erythromycin’.
If attribute X has value v then resource R must execute the activity.	Loans with value more than 1000000 Euro must only be approved by CFO.
If activity T_1 is done by agent A , activity T_2 must be done by the same agent.	A customer complain must be handseled with the same agent registered the customer request.

Table 1. Collection of data-aware and resource-aware compliance rules

4.1 Motivating Example

A process model, expressed in BPMN notation, describing a simplified procedure for procurement in a company is shown in Fig. 6. The process starts with activity *receive request for purchasing goods*. Afterwards the sub-process for *choosing supplier* is activated. Every purchase requisition carries a risk which is calculated based on a set of factors such as history of supplier's business, legal background, and etc. Consequently the risk is classified as *high* or *low*. If risk is high and not acceptable, the procurement expert investigates about *risk reduction measures*. These measures may lead to a *low* risk or the risk may stay still *high*. The procurement expert must decide based on the new information to continue with *risk reduction measures* or *accept the risk*. If risk is *accepted* *purchase order* can be *prepared*. For every *purchase order* two *approvals* are required by two different agents. An approved purchase order is *sent to supplier*. The process continues with *receiving invoice*. After *receiving goods*, the *payment* is done and the process terminates.

To prevent fraud, the company has defined various compliance rules. The employees may deviate from the modelled process as long as they do not violate the compliance rules. Here, we present just two of the rules that must be followed:

- *Rule 1* (Four-eye principle): *purchase order approval(I)* and *purchase order approval(II)* must be executed by two different agents.
- *Rule 2*: If *risk* of procurement from a supplier is calculated *high*, *risk reduction measures* must be executed at least once.

During auditing, the company checks if the procurement process were executed in compliance with relevant compliance rules including above mentioned ones. Therefore, executions of this process are recorded in terms of an event log and are used for compliance checking. These rules cannot be checked by solely considering the execution order of events; instead, specific event attribute values need to be taken into account.

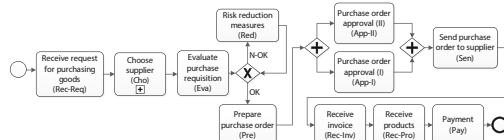


Fig. 6. A process model for procurement

	name	Rec-Req	Cho	Eva	Pre	App-I	App-II	Sen	Rec-Inv	Rec-Pro	Pay
resource	John	Luis	Luis	Sara	Arash	Clara	Luis	Luis	John	Luis	
role	clerk	expert	expert	clerk	director	manager	expert	expert	clerk	expert	
inventory level	500 pce.	300 pce.	—	—	—	—	—	—	—	—	—
risk	—	low	high								
time	20-Jan-14	21-Jan-14	22-Jan-14	24-Jan-14	26-Jan-14	26-Jan-14	28-Jan-14	28-Jan-14	29-Jan-14	29-Jan-14	29-Jan-14

Fig. 7. Trace σ recorded an instance of the procurement process execution

The synthetic event log L contains the executions of the procurement process for January of 2014. The trace σ shown in Fig. 7 is taken from the event log L .

4.2 Methodology

As discussed in 3, compliance rules fall into two main categories. The techniques we employ to check compliance of rules varies per category. Figure 8 depicts the approach we use for checking the compliance rules of the first category. For these rules, we need to check if a restricted activity was executed with a correct data or resource. Hence, the restriction on data attributes must be checked and the underlying control-flow is assumed to be correct. For this, as is shown in Fig 8, we prepare the log and enrich it with some necessary information in *steps 1*, and *2*. To check every compliance rule, we need to capture the scope of the rule i.e., we identify when a compliance rule is triggered. The scope of the rule is defined based on the occurrence of an activity or sequence of activities creating a control-flow alignment. In *step 3*, we specify the scope of the compliance rule in form of a classical Petri net and create a control-flow alignment in *step 4*. We generate a log using the alignment result and enrich it in *step 5* with diagnostics we obtained in the previous step. Later in *step 6*, we create a data-aware alignment using a data-aware Petri net to check if the data condition specified in the rule, holds.

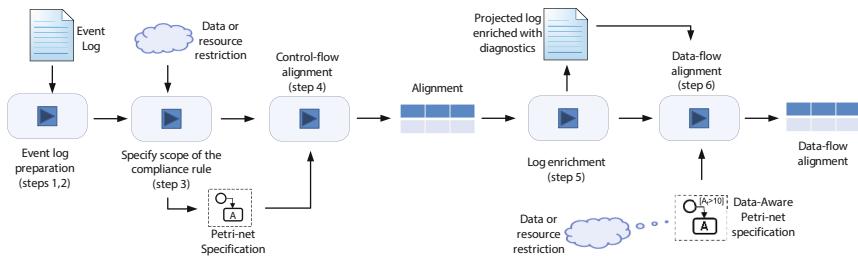


Fig. 8. Compliance checking approach for compliance rules restricting a data attribute

The compliance rules of the second category restrict the execution of activities when a certain data or resource condition holds. These rules assume the data-flow is correct and the control-flow i.e., the execution of activities under a certain data condition, must be checked. For this, we create a data-aware alignment to identify all the situations where a compliance rule must hold. Then creating a control-flow alignment we can check if activities were executed correctly under the specified data condition. Fig. 9 describes the approach we employ for checking compliance rules of the second category. In this approach after preparation of the event log in *steps 1*, *2*, and *3*, in *step 4* we apply data-aware alignment technique to identify all the situations where the compliance rule must hold. In *step 5* we enrich the log with diagnostics obtained in data-aware alignment. We define the scope of the compliance rule in *step 6* and apply control-flow alignment technique in *step 7* to check if activities were executed correctly under the specified data condition. In the following we will elaborate on both approaches by checking the sample compliance rules defined for the motivating example.

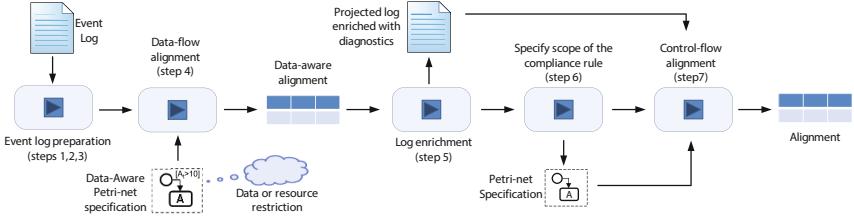


Fig. 9. Compliance checking approach for compliance rules restricting activities when a certain data or resource condition holds

4.3 Compliance Checking of Rules Restricting Data Attributes

From the compliance rules listed for the motivating example in Sect. 4.1, *Rule 1* falls in the first category of compliance rules. For checking such rules we employ the approach shown in Fig. 8. In the following we will explain the technique step by step for this example.

Compliance Rule 1 (four-eye principle): Purchase order approval(I) and purchase order approval(II) must be executed by two different agents.

Step 1. In this step we abstract the log from event attributes and their values which are not relevant for the rule. This rule restricts the data attribute *resource* of activities with $(name: App-I)$, and $(name: App-II)$. Therefore we can focus on event attributes *name*, and *resource* and discard other event attributes of σ . Consequently we obtain σ_{11} shown in Fig. 10.

L	name	Rec-Req	Cho	Eva	Pre	App-I	App-II	Sen	Rec-Inv	Rec-Pro	Pay
	resource	John	Luis	Luis	Sara	Arash	Clara	Luis	Luis	John	Luis

Fig. 10. Trace σ_{11} obtained from step 1

Step 2. In the second step we abstract σ_{11} further from information that is not relevant for checking *Rule 1*. Therefore we keep the value of data attribute *name* when $(name: App-I)$, and $(name: App-II)$ and replace all other values of attribute *name* with a generic value Ω . *Rule 1* restricts the value of data attribute *resource* at activities with $(name: App-I)$ and $(name: App-II)$; hence we keep the value of *resource* at these activities and discard its value at any other event. As a result of this step we obtain the trace σ_{12} shown in Fig. 11.

Step 3. The four-eye rule belongs to the first category (Fig. 8). Hence, the control-flow is assumed to be correct. For this rule activities with $(name: App-I)$, $(name: App-II)$ may be executed in any order. Considering this information, in this step we decide where the

L	name	Ω	Ω	Ω	Ω	App-I	App-II	Ω	Ω	Ω	Ω
	resource	-	-	-	-	Arash	Clara	-	-	-	-

Fig. 11. Trace σ_{12} obtained from step 2

compliance rule is triggered. As soon as any of the events with $(name: App-I)$ or $(name: App-II)$ occurs, the rule is activated.

Some compliance rules may hold several times in a trace. To elaborate more on this, consider the compliance rule stating : “*every time activity A occurs it must be followed by activity B*.”. If A occurs multiple times in a trace, we will have different instances of this rule in a trace and every occurrence of A must be followed by activity B. When deciding on the scope of a compliance rule, we need to consider if multiple instances of a compliance rule is allowed or not.

After defining the rule scope, we design a Petri net which describes the rule. For this, we can apply the *compliance elicitation* technique available in [15]. The Petri net describing *Rule 1* is shown in Fig. 12. This net starts by firing transition *Start* and a token in place *Final* represents a completed trace. The part between transitions I_{st} and I_{cmp} represents an instance of the compliance rule. Please note that several instances of the compliance rule are allowed and captured in the structure of the net. The rule becomes active when I_{st} fires and it ends when I_{cmp} fires. After the instantiation of every compliance rule, activities with $(name: App-I)$, and $(name: App-II)$ may occur in any order. The Ω transition represents any event in the trace with $name: \Omega$. The hollow transitions (*Start*, I_{st} , I_{cmp} , and *End*) are invisible.

Step 4: Having trace σ_{12} and the Petri net, we apply our control-flow alignment technique to capture the scope where the compliance rule must hold. This alignment indicates deviations of σ_{12} from the Petri net and diagnostics about the deviations. For control-flow compliance checking we ignore attributes other than *name* and use this attribute for mapping events and activities in aligning trace σ_{12} to the Petri net. γ_1^c in Fig. 13 shows the resulting alignment.

The alignment shows that trace σ_{12} can be replayed without any real violation on the Petri net. Please note that the missing events (\gg) indicated in the top row of the alignment are related to invisible transitions in the Petri net and are not considered as real violations i.e., costs of deviation for these events are 0.

Step 5: In this step we enrich trace σ_{12} with the additional information we obtained from the control-flow alignment as follows. We insert artificial events with attribute *name* and values (*Start*, I_{st} , I_{cmp} , *End*) in the trace where the alignment identified missing events (\gg). As a result we obtain trace σ_{15} shown in Fig. 14. This additional information marks

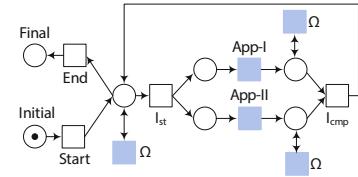


Fig. 12. Petri net specifying the scope of *Rule 1*

L	name	>>	Ω	Ω	Ω	Ω	>>	App-I	App-II	>>	Ω	Ω	Ω	Ω	>>
	resource	-	-	-	-	-		Arash	Clara		-	-	-	-	
S	name	Start	Ω	Ω	Ω	Ω	I-st	App-I	App-II	I-cmp	Ω	Ω	Ω	Ω	End

Fig. 13. Alignment γ_1^c obtained from step 4

L	name	Start	Ω	Ω	Ω	Ω	I-st	App-I	App-II	I-cmp	Ω	Ω	Ω	Ω	End
	resource	-	-	-	-	-	-	Arash	Clara	-	-	-	-	-	-

Fig. 14. Trace σ_{15} enriched with additional information obtained from control-flow alignment

the scope of the compliance rule in the trace and it shows that *Rule 1* was triggered once in the trace.

Step 6: In this step we apply the data-aware alignment technique on the enriched log (σ_{15}) obtained from step 5 and the extended Petri net with data annotations. Figure 15 shows the data-aware Petri net describing the restriction on data attribute *resource* w.r.t. *Rule 1*.

The data attribute *resource* is shown as the yellow eclipse in Fig. 15. The dotted line which associates attribute *resource* to transitions *App-I*, *App-II* indicates that the value of attribute *resource* can get updated by occurrence of any of these activities. As explained in Sect. 2, we define a variable *R* that takes the value of attribute *resource* whenever this attribute is updated by any of the transitions *App-I* or *App-II*. This is shown by the annotation $\{W : r = e.resource\}$ on these two transitions. Please note that we annotate the transition *I_{st}* with $\{W : r = \text{undefined}\}$ to express that as soon as the compliance rule is activated the value of variable *r* is set to *undefined*. The restriction on data attribute *resource* which is specified in *Rule 1* is expressed by the guard $[r = \text{undefined} \mid\mid r' \neq e.resource]$ annotated at transition *App-I* and *App-II*. This guard specifies that the variable *r* is allowed to get a new value if its previous value is *undefined* [$r = \text{undefined}$] or in case it has already a value, the new value must be different with current value [$r' \neq e.resource$]; implying activities *App-I* and *App-II* must be executed with different *resources*. The data-aware alignment of σ_{15} against the data-aware Petri net is shown in Fig. 16. The alignment illustrates that both guards are evaluated to correct. Hence the trace is compliant with *Rule 1*.

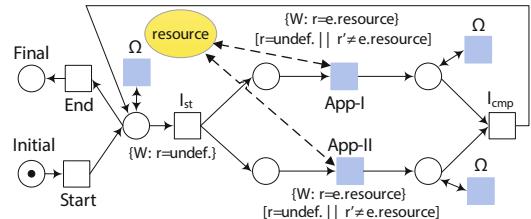


Fig. 15. Data-aware Petri net of *Rule 1*

4.4 Compliance Checking of Rules Restricting Activities When a Certain Data Condition Holds

The compliance *Rule 2* listed for the motivating example in Sect. 4.1 falls in the second category of compliance rules. For checking these rules we employ the approach shown in Fig. 9. Next we will explain this technique step by step for *Rule 2*.

Compliance Rule 2: If risk of procurement from a supplier is calculated high, risk reduction measures must be executed at least once.

L	name	Start	Ω	Ω	Ω	Ω	I-st	App-I	App-II	I-cmp	Ω	Ω	Ω	End
resource	—	—	—	—	—	—	Arash	Clara	—	—	—	—	—	—
S	name	Start	Ω	Ω	Ω	Ω	I-st	App-I	App-II	I-cmp	Ω	Ω	Ω	End
resource	—	—	—	—	—	—	Arash	Clara	—	—	—	—	—	—

Fig. 16. Data-aware alignment γ_1^d obtained from step 6

Step 1. Similar to the log preparation procedure we followed in the previous approach, we first abstract the log from event attributes and their values which are not relevant for the rule. Therefore we remove other attributes apart from *name* and *risk* from the trace and we obtain σ_{21} in Fig. 17.

L	name	Rec-Req	Cho	Eva	Pre	App-I	App-II	Sen	Rec-Inv	Rec-Pro	Pay
	risk	—	low	high	high	high	high	high	high	high	high

Fig. 17. Trace σ_{21} obtained from step 1

L	name	Ω									
	risk	—	low	high							
	data condition	Ω	risk								
	write	write	write	write	write	write	write	write	write	write	write

Fig. 18. Traces σ_{22} , and σ_{23} obtained respectively from steps 1, and 2

Step 2. We keep the value of event attribute *name* for *risk reduction measure* and substitute all other values of attribute *name* with the generic value Ω . In addition we keep the attribute *risk* with all its values. Trace σ_{22} shown in Fig. 18 is the result of step 2. As is shown in σ_{22} , activity *risk reduction measure* was never executed.

Step 3. In Rule 2, the restriction is on the number of occurrences of activity *risk reduction measures* when data attribute *risk* has value *high*, hence the rule is triggered as soon as attribute *risk* gets value *high*. Not all the events have access to attribute *risk* i.e., not all events can *update* the value of attribute *risk*. Therefore, we need to capture the existence and changes in the value of *risk*. To this end we introduce a new event attribute named *data condition* which gets the value *risk write* whenever an event records a value for its attribute *risk*. The *data condition* gets the value Ω if *risk* doesn't have any value in an event. Trace σ_{23} shown in Fig. 18 is obtained from step 3.

Step 4. As discussed in Sect. 2, we express data and resource-aware compliance rules using a data-aware Petri net. The corresponding alignments of a data-aware Petri net to a log will then allow to check these rules. For the alignment we use the abstracted and enriched log σ_{23} obtained from step 3.

The data-aware Petri net shown in Fig. 19 has two transitions Ω and *risk write* where we map them respectively to the values of event attribute *data condition* in the log. Therefore, events with (*data condition*:*risk write*) are mapped to transition *risk write* in the data aware Petri net and events with (*data condition*: Ω) are mapped to the transition labelled Ω . The other event attributes *name* and *risk* are mapped respectively to data attributes *name* and *risk*.

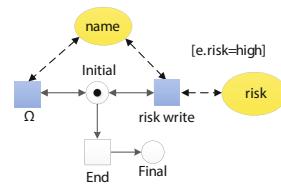


Fig. 19. The data-aware Petri net of Rule 2

We would like to capture all situations where the data condition of *Rule 2* holds. That is, we want to capture the events where *risk* is *high* and check if such an event is followed at least once by the activity *risk reduction measure*. Hence the transition *risk write* in the data-aware Petri net (Fig. 19) is guarded with $[e.risk = \text{high}]$.

We check the conformance of the trace σ_{23} against the data-aware Petri net of *Rule 2* applying data-aware alignment. The data-aware alignment technique checks if events with *data condition:risk write* were executed with the admissible value (*high*) for its *risk* attribute. As a result the data-aware conformance checking technique will return a data-aware alignment γ_2^d as is shown in Fig. 20. The columns colored in red indicate the events with *(data condition:risk write)* which didn't have the value *high* for attribute *risk* (yellow eclipses in the data-aware Petri net).

	name	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω
L	risk		low	high							
	data condition	Ω	risk write								
	name	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω
S	risk		high								
	data condition	Ω	risk write								
	name	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω

Fig. 20. Data alignment γ_2^d of trace σ_{23} for compliance *Rule 2*

Step 5. In this step we enrich trace σ_{23} with diagnostics we obtained from the data-aware alignment in the previous step as follows:

i) We insert an event attribute *condition holds* to all events in the trace. This new event attribute gets no value (\gg) if no value is recorded for the attribute *risk* of the events. Whenever a deviation is indicated in the alignment (columns colored in red), *condition holds* gets value *false*, else it gets the value *true*. Note that value *false* for attribute *condition holds* indicates all the situations in the trace where *risk* was not *high*.

ii) We insert event attribute *admissible risk* to the violating events. This new attribute shows the admissible value; for our example (*admissible risk:high*).

iii) We insert event attribute *combined name&condition* to all events. The value of this attribute for every event is the combination of the value for the corresponding attributes *name* and *condition holds* of that event. This event attribute is added to enable us in the next step to check if an activity was executed with the right data or resource.

As a result of above mentioned modifications, we obtain trace σ_{24} (shown in Fig. 21) enriched with diagnostics and added information.

Step 6. In this step we define the scope of the compliance rule. *Rule 2* is triggered whenever *risk* is calculated *high* and it is satisfied as soon as activity *risk reduction measure* is executed. Please note that afterwards the *risk* may stay *high* or *risk reduction measure* may occur arbitrary number of times. If we interpret this rule in context of the trace σ_{24} , we can rephrase the rule to: “occurrence of an event with *(combined name&condition: $\Omega - \text{true}$)* must be followed at least once by an event having value *risk*

	name	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω
	risk		low	high							
L	data condition	Ω	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write
	condition holds	>>	false	true							
	admissible risk	>>	high	high	high	high	high	high	high	high	high
	combined name& condition	$\Omega \rightarrow \Omega$	$\Omega \text{-false}$	$\Omega \text{-true}$							

Fig. 21. Trace σ_{24} enriched with diagnostics obtained from data-aware alignment

reduction measures-true or risk reduction measures-false for event attribute *combined name&condition*. In other words, an event with $\Omega - \text{true}$ indicates that first activity which calculated (*risk:high*) is executed. In addition any of the events with *risk reduction measures-true* or *risk reduction measures-false* indicates that activity *risk reduction measures* was executed.

When defining the scope of *Rule 2*, we need to consider whether multiple instances of the rule are allowed. In case of *Rule 2*, only fulfilling one instance of the rule is enough and after fulfilling it once, it won't be triggered again although *risk* may still be *high*. After deciding about the scope of the rule we can design a Petri net which describes the rule. The result is shown in Fig. 22.

Step 7. Having trace σ_{24} and the Petri net specification of in Fig. 22, we apply control-flow conformance checking to get an alignment between σ_{24} and the specification. This alignment indicates deviations and diagnostics about the deviations. For control-flow alignment we ignore attributes other than '*combined name&condition*' and align the trace σ_{24} to the net shown in Fig. 22. γ_2^c in Fig. 23 shows the alignment obtained.

The alignment indicates a deviation in trace σ_{24} implying that *risk* has been *high*, hence activity *risk reduction measure* should have happened but it didn't.

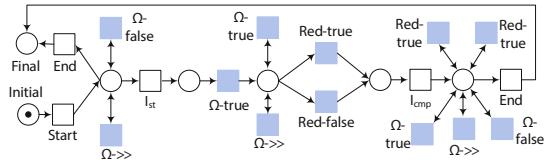


Fig. 22. Petri net specifying compliance *Rule 2*

5 Implementation

The presented technique is supported by two novel plugins in the *ComplianceFramework* package of ProM 6.3¹. The *Data and Resource Compliance Checking-Rules Restricting Activities* plug-in checks the compliance of event logs against rules of the first category

¹ Available from www.processmining.org

		Ω	Ω		Ω											
	risk		low		high											
L	data condition		risk write		risk write											
	condition holds	>>	false		true											
	admissible risk	>>	high		high											
	combined name& condition	$\Omega\rightarrow\Omega$	$\Omega\text{-false}$		$\Omega\text{-true}$											
S	combined name& condition	Start	$\Omega\rightarrow\Omega$	$\Omega\text{-false}$	I-st	$\Omega\text{-true}$	Red-true	I-cmp	End							

Fig. 23. Data-aware alignment γ_2^c obtained from step 7

and takes an event log and a data-aware Petri net as input. The *Data and Resource Compliance Checking-Rules Restricting Attributes* checks the compliance of event logs against rules of the second category and takes an event log, a data-aware Petri net, and a classical Petri net as input. Both return diagnostics about deviations in the form of an alignment. The resulting alignments provide diagnostics by showing data and resource violations and control-flow violations projected over the original log.

6 Experimental Results

We applied our approach and toolset in two case studies: one for checking compliance of a personal loan process within a global financial organization and one for analyzing the process of patient treatment in ICU department of a Dutch hospital against a medical guideline.

Case study 1: The event log related to the first case study is taken from BPI challenge of 2012² and it has 13.087 traces. A compliance rule restricting this process states: “loan applications with requesting amount less than 5000 and more than 50000 must not be approved”. This compliance rule is of second type of data-aware compliance rules. We found 117 number of deviations from the compliance rule out of all 13.087 traces executed, i.e., 117 approved application requested an amount for more than 50000 or less than 5000.

Case study 2: In this case study we investigated the compliance of an event log taken from ICU department of a Dutch hospital with a medical guideline restricting tube feeding nutrition of patients. The event log has 1207 traces; each trace recorded the treatment that a patient received in ICU department. The compliance requirement states: “If gastric tube feeding cannot be increased then use *Domperidone* or *Metoclopramide*. The starting dosage is usually (12 · 50ml) and it is recommended that the increase follows the pattern (12 · 50ml, 12 · 100ml, 12 · 120ml)”.

² This event log is publicly available at:

dx.doi:10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

The guideline is not precise here, hence we check two different data-aware rules: *i)* The nutrition must increase, else *Domperidone* or *Metoclopramide* must be administered to the patient. *ii)* the increase must follow the pattern (12 · 50ml, 12 · 100ml, 12 · 120ml).

We first checked if the nutrition has been increased for respective patients and if not, did they receive either *Domperidone* or *Metoclopramide*. We observed that from 1207 patients treated in ICU, 209 received tube feeding nutrition. In addition we found that *Metoclopramide* has not been administered for these patient; only one patient has received this medicine and it has been independent from tube feeding nutrition. For 72 patients, the tube feeding nutrition has increased without any problem. 56 patients received *Domperidone* when nutrition was not increased.

We observed in total 81 violating traces. We identified several patterns for these violations. In some of violating traces, we observed that although nutrition has increased, patients received *Domperidone*. It could be that that *Domperidone* was administered to these patients independent from nutrition and for other purposes.

Another group of violations are related to patients that did not receive *Domperidone* although nutrition was not increased for them. We checked these traces further and found that this violation occurs in two situations. One group contained *real violations* because patients never received *Domperidone* despite the nutrition was not increased. However for another group of patients we observed several iterations of tube feeding nutrition with an increase inside every iteration. For example a patient has received nutrition following two times the pattern (12 · 50ml, 12 · 100ml, 12 · 120ml). That is we see the increase in every occurrence of the pattern, yet the second occurrence of the pattern starts again from (12 · 50ml).

We also investigated if the recommended pattern of increase has been followed or not. We found out that only for less than 20% of patients the recommended pattern of the guideline is followed. From the remaining we identified 3 groups of patients. One group of course were the 56 patients that the nutrition was not increased for them. For the second group of patients, the nutrition was increased with the pattern (12 · 50ml, 12 · 100ml, 12 · 140ml). The third group mostly followed the pattern (500ml, 1000ml, 1500ml, 1700ml). These patients received *Domperidone* once in every 24 hours, unlike the other groups that received *Domperidone* every two hours a day.

Applying the technique presented in this paper, we were able to check compliance of all the traces in the event logs rather than being limited to sample based compliance checking. The technique works on large event logs because we can focus on events relevant to a specific compliance rule and abstract from all other events. Our technique identified and located the deviations by visualizing for each trace the difference between admissible and observed behavior in the alignment. The extent of the deviation is reported in two ways: (1) The alignment visualizes observed and expected values of deviating events. (2) We compute statically information about the number of deviations per case, in total, etc. A domain specialists can use this information to assess their severity and analyze root-causes.

7 Related Work

Existing work in data and resource compliance checking mainly focuses on design time verification. In [6], authors incorporate data in specification of compliance rules. These

rules are expressed using a query language which is an extended version of BPMN-Q and they are formalized by mapping into PLTL. They can visualize violations by indicating execution paths in a process model causing them.

In [17] the control-flow is modeled and object life-cycles model other perspectives. Later object state change becomes explicit in the process model and then the process model is used to generate life-cycles for each object type used in the process. The consistency between business process models and life-cycles of business objects is checked. Apart from the fact that this approach also focuses on verification of models, it is not discussed how deviation points represented and if further diagnostics are provided.

Other approaches such as [9,19,21] enforce compliance of business processes through the use of compliance patterns. These patterns include some of the data and resource-aware compliance requirements such as four-eye principle. However, specific checking technique are not discussed. Further on, the work in [10] addresses the verification of process models for data-aware compliance requirements. The authors do not apply a particular modeling language for specifying compliance rules, but introduce a general notion for data-aware conditions. For checking data-aware requirements, abstraction strategies are introduced to reduce complexity of data-aware compliance checking and deal with state explosion issues. This is achieved by abstracting from concrete state of data objects. The approach automatically derives an abstract process model and an abstract compliance rule.

Process mining techniques [1] offer a means to more rigorously check compliance and ascertain the validity of information about an organization's core processes. The challenge is to compare the prescribed behavior (e.g., a process model or set of rules) to observed behavior (e.g., audit trails, workflow logs, transaction logs, message logs, and databases). Various techniques have been proposed for checking control-flow compliance rules based on event data including LTL-based checking [3]. In [13] both LTL-based and SCIFF-based (i.e., abductive logic programming) approaches are used to check compliance with respect to a declarative process model and an event log. Dozens of approaches have been proposed to check conformance given a Petri-net and an event log [2,16].

The classical data-aware conformance checking [11] that we have applied in our approach, allows for aligning event logs and process models for multi-perspective conformance checking but it has some limitations; as we can only check the compliance rules of the first category we discussed earlier. In addition if a deviation is observed for an activity that is restricted with several data attributes at the same time, the classical data-aware conformance checking can only indicate the deviation but not the specific data attribute(s) causing the deviation; resulting in less precise diagnostics. Moreover using classical data-aware conformance checking, we cannot focus only on specific rules and abstracting from other activities that are not restricted by respective compliance rule. That is we need to provide a data-aware Petri net that captures the behavior of the whole process; consequently make the checking less flexible. We have covered above mentioned limitations in our approach by extending the classical data-aware conformance checking.

8 Conclusion and Future Work

In this paper, we provided an approach for data and resource-aware compliance checking of behavior recorded in execution logs. We show a collection of different compliance constraints restricting process data and resources. In addition, we provide two generic techniques for checking these rules based on alignments. Our technique separates control-flow, data and resource compliance checking to the possible extent, and provides integrated diagnostic information about both control-flow violations, and data and resource related compliance violations. In particular, our technique is capable of showing diagnostic information about violations of a compliance rule in a process instance. We have shown this technique to be feasible for compliance rules on data dependencies between two or three data attributes. More complex rules are possible but require additional pre-processing and data-aware alignment steps. A more scalable technique is subject to further research.

We provide an implementation of our techniques in the *ComplianceFramework* package of ProM. The software has been tested in synthetic logs and two case studies involving real-life logs from a financial institute and ICU department of a hospital. The results are encouraging: we were able to uncover various violations and no performance issues were encountered. Future research aims at making the approach more user-friendly. While the compliance checking itself is fully automatic, the user still has to create formal compliance rules and interpret the results. Formal data-aware compliance rules could be elicited in questionnaire-based approach similar to [15]. Further, a higher-level compliance language and a dashboard for integrating results of control-flow, temporal, and data and resource-aware compliance checking to provide a multi-perspective view on data are required.

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes, pp. 1–352. Springer (2011)
2. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replay history on process models for conformance checking and performance analysis. Wiley Interdisc. Rev.: Data Mining and Knowledge Discovery 2(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.) OTM 2005. LNCS, vol. 3760, pp. 130–147. Springer, Heidelberg (2005)
4. Accorsi, R., Stocker, T.: On the exploitation of process mining for security audits: the conformance checking case. In: SAC, pp. 1709–1716. ACM (2012)
5. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: EDOC, pp. 55–64. IEEE Computer Society (2011)
6. Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for data aware compliance rules. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 500–515. Springer, Heidelberg (2009)
7. Botha, R.A., Eloff, J.H.P.: Separation of duties for access control enforcement in workflow environments. IBM Systems Journal 40(3), 666–682 (2001)

8. Elgammal, A., Turetken, O., van den Heuvel, W.-J., Papazoglou, M.: On the formal specification of regulatory compliance: A comparative analysis. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 27–38. Springer, Heidelberg (2011)
9. Elgammal, A., Turetken, O., van den Heuvel, W.-J., Papazoglou, M.: Root-cause analysis of design-time compliance violations on the basis of property patterns. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 17–31. Springer, Heidelberg (2010)
10. Knuplesch, D., Ly, L.T., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On enabling data-aware compliance checking of business process models. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 332–346. Springer, Heidelberg (2010)
11. de Leoni, M., van der Aalst, W.M.P.: Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 113–129. Springer, Heidelberg (2013)
12. de Leoni, M., Maggi, F.M., van der Aalst, W.M.P.: Aligning event logs and declarative process models for conformance checking. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 82–97. Springer, Heidelberg (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. TWEB 4(1) (2010)
14. 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)
15. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Supporting domain experts to select and configure precise compliance rules. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013 Workshops. LNBP, vol. 171, pp. 498–512. Springer, Heidelberg (2014)
16. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)
17. Ryndina, K., Küster, J.M., Gall, H.: Consistency of business process models and object life cycles. In: Kühne, T. (ed.) MODELS 2006. LNCS, vol. 4364, pp. 80–90. Springer, Heidelberg (2007)
18. Samarati, P., di Vimercati, S.D.C.: Access control: Policies, models, and mechanisms. In: Focardi, R., Gorrieri, R. (eds.) FOSAD 2000. LNCS, vol. 2171, pp. 137–196. Springer, Heidelberg (2001)
19. Schumm, D., Turetken, O., Kokash, N., Elgammal, A., Leymann, F., van den Heuvel, W.-J.: Business process compliance through reusable units of compliant processes. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 325–337. Springer, Heidelberg (2010)
20. Ramezani Taghiabadi, E., Fahland, D., van Dongen, B.F., van der Aalst, W.M.P.: Diagnostic information for compliance checking of temporal compliance requirements. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 304–320. Springer, Heidelberg (2013)
21. Türetken, O., Elgammal, A., van den Heuvel, W.J., Papazoglou, M.P.: Enforcing compliance on business processes through the use of patterns. In: ECIS (2011)
22. di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Access control: principles and solutions. Softw., Pract. Exper. 33(5), 397–421 (2003)

RelBOSS: A Relationship-Aware Access Control Framework for Software Services

A.S.M. Kayes, Jun Han, Alan Colman, and Md. Saiful Islam

Swinburne University of Technology, Victoria 3122, Australia
`{akayes,jhan,acolman,mdsaifulislam}@swin.edu.au`

Abstract. *Context-awareness* is an important aspect of the *dynamically changing environments* and the *relationship context information* brings new benefits to the access control systems. Existing relationship-aware access control approaches are highly domain-specific and consider the expression of access control policies in terms of the relationship context information. However, these approaches are unable to dynamically capture the granularity levels and strengths of the relevant relationship. To this end, in this paper we present a formal *Relationship-Aware Access Control (RAAC)* model for specifying the relevant *relationship context information* and the corresponding *access control policies*. Using the *RAAC* model, we introduce an ontology-based framework, *Relationship-Based access control Ontology for Software Services (RelBOSS)*. One of the main novelties of the framework is that it dynamically captures the *relationship context information* (the *type/name*, *granularity levels* and *strengths* of the relevant relationship). Experiments with a software prototype confirm the feasibility of our framework.

Keywords: Relationship-aware service, Relationship context information, Relationship-aware access control, Access control policy.

1 Introduction

The rapid advancement of computing technologies [1][2] has led the world to a new paradigm of access control from static to dynamic context-aware environments [3]. Such a paradigm shift brings with it lots of opportunities and challenges. On the one hand, users demand access to software services in which the relationship information need to be explicitly/implicitly captured for the purpose of access control decision making. On the other hand, such access needs to be carefully controlled due to the additional challenges of capturing the granularity levels and strengths of that relationship. Consider a software service in a healthcare domain, a doctor requests a patient's electronic health records. This access request should be granted based on the relevant relationship between the doctor and the corresponding patient, e.g., only a treating doctor can access the patient's private health records. Such service access request can be realized by modeling the relationship with certain granularity levels (e.g., 'TreatingDoctor-Patient' or 'Doctor-Patient') and strengths ('strong' or 'weak') from the available context information. The above challenges require a new *relationship model* for

capturing the *relationship context information* and a *policy model* for making *relationship-aware access control decisions*.

Existing context definitions describe the information which characterizes the states of the entities [4][5][6]. As proposed by Dey [5], “*context is any information that can be used to characterize the situation of an entity (an entity may be a person, place or object)*”. However, other than the information about entities, the information about relationship between relevant entities (relationship context information) is not considered in this context-awareness research.

The relationship-aware access control approaches (e.g., [1], [7], [8]) consider the expression of access control policies in terms of the relationship information. These approaches only consider the *person-centric* or interpersonal relationships to control access to resources or services. However, the different granularity levels and strengths of the relevant relationship are not considered in these works. Carminati et al [1] consider the relationships between users (e.g., *friend*, *friend-of-friend*) in the access control processes. Zhang et al [7] consider the access permissions as binary relations between subjects and objects, and ReBAC [8] considers the relationships between individual users (e.g., *professional association*) in the access control policies. Different from these approaches, our approach represents much richer forms of relationships among relevant entities with different granularity levels and strengths of that relationship, and expresses the access control policies in terms of this relationship context information.

The Contributions. In this paper we present a framework, *RelBOSS*, in order to provide relationship-aware access to information resources and software services. The novel features of this framework are as follows:

- (C1) *Relationship Model.* Our framework uses the *relationship context information* to provide relationship-aware access to software services (authorization), where we present a *relationship model* to represent and reason about the relevant relationship with different granularity levels and strengths.
- (C2) *Relationship-Aware Access Control Policy Model.* Our framework presents a *policy model* to specify relationship-aware access control policies. The policy model supports access control to the appropriate software services based on the relevant relationship context information.
- (C3) *Ontology-Based Framework Implementation.* Based on the relationship and policy models, we introduce an *ontology-based framework* for identifying relationship context information, and enforcing relationship-aware access control policies. Our ontology-based framework represents the basic context entities and information using the ontology language OWL, extended with SWRL for identifying and reasoning about relevant relationship context information and the corresponding access control policies.
- (C4) *Prototype Implementation and Evaluation.* We have presented a *software prototype* for the development of the relationship-aware access control applications and demonstrated the effectiveness of our framework through a healthcare *case study*. To demonstrate the feasibility of our framework, we have conducted a number of experiments on a simulated healthcare envi-

ronment. We have quantified the *performance overhead* of our framework for measuring the response time.

Paper Outline. The rest of the paper is organized as follows. Section 2 presents a healthcare application scenario to motivate our work. Next, we present a formal definition of our framework, a relationship model to specify relationship context information and a policy model for specifying relationship-aware access control policies in Section 3. Section 4 presents an ontology-based development platform for our framework. Next, we describe the prototype implementation along with the viability of the framework in Section 5. After that, we discuss the related work in Section 6. Finally, Section 7 concludes the paper.

2 Research Motivation and General Requirements

In this section, we first outline a motivating scenario that illustrates the need for the incorporation of relationship context information in the access control processes. We then distill the requirements for managing the access to software services in a relationship-aware manner.

Motivating Scenario. *The scenario begins with patient Bob who is in the emergency room due to a heart attack. Jane, a medical practitioner of the hospital, is required to treat Bob and needs to access Bob's electronic health records (e.g., emergency medical records, private health records) in an emergency health condition. However, in a normal health condition, only the treating practitioners should be able to access the patient's private health records, if he/she is present in the same location with patient.*

The relationship information describing the context entities relevant to access control in the above scenario includes: the *type/name* of the relationship between the service requester and the service owner (e.g., 'User-Owner' relationship), the different *granularity levels* of the relevant relationship (e.g., 'User-Owner' relationship, at granularity level 0 (i.e., 'Doctor-Patient' relationship), 'User-Owner' relationship, at granularity level 1 (i.e., 'TreatingDoctor-Patient' or 'EmergencyDoctor-Patient' relationship), the *strengths* of the relevant relationship (e.g., the *strengths* of the 'User-Owner' relationship is 'strong' or 'weak')). This relationship context information needs to be identified/inferred based on the currently available context information using user-defined rules.

General Requirements. To support such relationship-aware access control in a computer application like an electronic health records management system, we need to consider the 3Ws: *who* (the appropriate users/roles) wants to access *what* (the appropriate software services), and *when* (the relationship context information). As different types of relationship context information are integrated into the access control processes, some important issues arise. These issues and their related requirements are as follows:

- (R1) *Representation of relationship context information:* What access control-specific elementary information should be identified as part of building a

relationship model specific to relationship-aware access control? Furthermore, how to represent and reason about the relevant relationship context information based on the currently available context information?

- (R2) *Specification of relationship-aware access control policies:* How to define the access control policies based on the relevant relationship context information to realize a flexible and dynamic access control scheme?
- (R3) *Implementation framework:* How to realize the relevant relationship context information and the corresponding relationship-aware access control policies in an effective way, in order to access/manage software services?

3 Relationship-Aware Access Control

In this section, we present our conceptual Relationship-Aware Access Control (RAAC) framework for Software Services.

In our previous work [9], we define the *context information* in which the two parts relevant to making access control decisions are: *the information characterizing the entities* and *the information characterizing the relationships between different entities*. In this paper, we elaborate the second part of the proposed context definition. Our main focus is to capture/infer the fine-grained relationship (the type/name, granularity levels and strengths) between different person-centric entities (*interpersonal relationship*), in order to control access to services depending on the relevant relationship context information.

Definition 1 (Relationship Context Information). The *relationship context information* used in an access control decision is defined as the relevant relationship (type/name) with different granularity levels and strengths of that relationship. The relationship *type*, *granularity levels*, and *strengths* are domain-dependent concepts, and their values can be obtained (captured/inferred) based on the access request (e.g., from the sensed contexts, inferred contexts, etc.).

The relationship context information ‘*Rel*’ can be formally defined as a tuple in the form of

$$\begin{aligned} Rel &= \{rel_1, rel_2, \dots, rel_n\} \\ rel &= <rel.name, rel.gLevel, rel.strength> \end{aligned} \quad (1)$$

where:

- *rel.name* denotes a relationship type, e.g., ‘*user-owner*’ relationship.
- *rel.gLevel* denotes the different granularity levels of the relationship, e.g., ‘*user-owner*’ relationship at granularity level 1, i.e., ‘*treatingDoctor-patient*’ or ‘*emergencyDoctor-patient*’ relationship.
- *rel.strength* denotes the relationship strengths of the relationship (‘strong’ or ‘weak’ relationship), e.g., the strengths of the relationship between doctor and patient is ‘strong’ relationship.

How to capture/represent the *type* of the relevant relationship, and how to identify/infer the different *granularity levels* and *strengths* of that relationship are discussed in the following sub-sections.

3.1 Representation of Relationship

A *relationship type* used in an access control decision can be captured based on the currently available context information.

Example 1: Concerning our application scenario, a relevant access control policy is as follows: a user Jane, who is a medical practitioner, by playing an emergency doctor (ED) role, is allowed to access ('read' or 'write' operation) the patient's emergency medical records (EMR). The following rule (2) is used to identify that the relationship *type/name* is "user-owner".

$$\begin{aligned} User(u) \wedge Owner(o) \wedge Resource(res) \wedge Relationship(rel_1) \wedge requests(u, \\ res) \wedge equal(res, "EMR") \wedge owns(res, o) \wedge hasRelationship(u, rel_1) \\ \wedge hasRelationship(o, rel_1) \wedge RelationshipName(rN) \quad (2) \\ \wedge has(rel_1, rN) \rightarrow rel.name(rN, "user - owner"). \end{aligned}$$

3.2 Reasoning about Relationship

The process of inferring the *relationship granularity levels* and *strengths* from the currently available context information is referred to *reasoning about relationship*. One of the main advantages of our framework to relationship-awareness is its reasoning capability; that is, once facts about the world have been stated, other facts can be inferred using an inference engine through the reasoning rules.

The *relationship granularity levels* and *strengths* can be inferred by performing logical composition on the relevant context information.

Example 2: Consider the policy mentioned in *Example 1*, the *granularity levels* of the relationship between user and owner can be inferred using the following rule (3) (i.e., a user by playing the 'ED' (emergency doctor) role can access a patient's medical records, when the relationship granularity is "emergency doctor-patient").

$$\begin{aligned} User(u) \wedge Role(r) \wedge plays(u, r) \wedge Relationship(rel_1) \\ \wedge equal(r, "ED") \wedge RelationshipGranularity(rG) \quad (3) \\ \wedge has(rel_1, rG) \rightarrow rel.gLevel(rG, 1). \end{aligned}$$

The above rule (3) identified the *granularity levels* of the relationship between resource requester and resource owner, which is 1 (i.e., "emergencyDoctor-patient" relationship). If the user in a doctor role, then the *granularity levels* of the relationship is 0 (i.e., "doctor-patient" relationship).

Example 3: Consider the same policy mentioned in *Example 1*, the *strengths* of the relationship between user and owner can be inferred using the rule (4).

$$\begin{aligned} User(u) \wedge Owner(o) \wedge Relationship(rel_1) \wedge RelationshipStrength(rs) \\ \wedge userIdentity(u, uID) \wedge connectedPeopleIdentity(o, cpID) \quad (4) \\ \wedge has(rel_1, rs) \wedge equal(uID, cpID) \rightarrow rel.strength(rs, "strong"). \end{aligned}$$

The above rule (4) identified the *strengths* of the relationship between resource requester (doctor) and resource owner (patient), which is “strong” (i.e., the physician is assigned as a treating doctor of the patient). If the physician is not a treating doctor of the patient, then the relationship strength is “weak”.

As a whole, the relevant relationship context information rel_1 between resource requester and resource owner (relationship *name* is ‘user-owner’) associated with the above-mentioned policy is represented as:

$$rel_1 = < "user - owner", 1, "strong" > \quad (5)$$

3.3 The RAAC Policy Model for Software Services

Based on the formalization of the traditional role-based access control (RBAC) model in [10], we present a formal definition of our relationship-aware access control (RAAC) policy model. Our policy model for RAAC applications extends the RBAC with relevant relationship context information. Our goal in this research is to provide a way in which the role-permission assignment policies in RBAC [10] that can be specified by incorporating relationship context information as policy constraints.

Definition 2 (RAAC Policy Model). Our relationship-aware access control policy model can be formally described as a tuple, where R , Rel , Ser , and $RAACP$ represents Roles, Relationship Context Information, Services, and Policies, respectively (see Formula (6)):

$$M_{RAAC} = (R, Rel, Ser, RAACP) \quad (6)$$

- **Roles (R):** R represents a set of roles that reflect user’s job functions or job titles within the organization (e.g., healthcare domain). The users or service requesters can play the roles and they are human-beings, whose service access requests are being controlled.

$$R = \{r_1, r_2, \dots, r_m\} \quad (7)$$

- **Relationship (Rel):** Rel represents the relationship context information in order to describe the relationship-aware access control policies.

$$Rel = \{rel_1, rel_2, \dots, rel_n\} \quad (8)$$

- **Services (Ser):** In this paper, we consider the resource (e.g., patients’ medical health records) in a service-oriented manner. A service request ser can be seen as a pair $\langle res, op \rangle$, $ser = \{(res, op) | res \in Res, op \in OP\}$, where Res represents a set of resources, $Res = \{res_1, res_2, \dots, res_q\}$ and OP represents a set of operations on the resources, $OP = \{op_1, op_2, \dots, op_r\}$. For example, the write operation on the emergency medical records is defined as $\langle EMR, write \rangle$. In this way, the fine-grained access control to the appropriate parts of a resource by the appropriate users can be realized.

$$Ser = \{ser_1, ser_2, \dots, ser_o\} \quad (9)$$

– **Policies (RAACP)**: RAACP represents the relationship-aware access control policies. Our RAAC policy model has relationship-aware role-service assignment policies to provide relationship-aware access to software services. In our RAAC policy, a service is considered as a software entity, in order to provide fine-grained access control and grant the right operation (e.g., particular mode of access, such as ‘read’ and/or ‘write’) to the appropriate resources (e.g., private health records, emergency medical records).

$$RAACP = \{rp_1, rp_2, \dots, rp_p\} \quad (10)$$

Our RAAC policy model extends the concept of role-permission assignments (*RPA*) in RBAC ($RPA \subseteq R \times P$) [10], by introducing the concept of relationship context information, called relationship-aware role-service assignments.

$$\begin{aligned} RAACP = & \{(r_1, rel_1, ser_1), ((r_2, rel_2, ser_2), \\ & \dots, ((r_m, rel_n, ser_o)\} \subseteq R \times Rel \times Ser \end{aligned} \quad (11)$$

Table 1. An Example of Relationship-Aware Access Control Policy

<p>If</p> $\begin{aligned} & RAACP(rp_1) \wedge User(u_1) \wedge hasUser(rp_1, u_1) \wedge Role(r_1) \wedge plays(u_1, r_1) \\ & \wedge hasRole(rp_1, r_1) \wedge equal(r_1, "ED") \wedge Service(ser_1) \wedge hasService(rp_1, ser_1) \\ & \wedge equal(ser_1, "writeEMR()") \wedge Relationship(rel_1) \wedge hasRelationship(rp_1, rel_1) \\ & \wedge rel.name(rel_1, nM) \wedge equal(nM, "user-owner") \wedge rel.gLevel(rel_1, gL) \\ & \wedge equal(gL, 1) \wedge rel.strength(rel_1, sT) \wedge equal(sT, "strong") \end{aligned}$ <p>Then</p> $canInvoke(u_1, ser_1)$

Example 4: Based on our policy model ($Role(r_1) \wedge Relationship(rel_1) \wedge Service(ser_1) \rightarrow (r_1, rel_1, ser_1) \in RAACP$), the above-mentioned rule (shown in Table 1) expresses the RAAC policy mentioned in Example 1, i.e., a User ‘ u_1 ’ by playing the Role ‘ r_1 ’ (emergency doctor role “ED”) can invoke the Service ‘ ser_1 ’ ($<EMR, write>$ or “ $writeEMR()$ ” service), if a relationship ‘ rel_1 ’ (named “*user-owner*” ($rel.name$), with “*emergencyDoctor-patient*” granularity ($rel.gLevel$ is 1) and “strong” strength ($rel.strength$)) is satisfied.

The identification of the relevant information to represent the *relationship context information* and specify the corresponding *RAAC policies* satisfies requirements R(1) and R(2), which is discussed earlier. To meet requirement R(3), we in the next section propose an *ontology-based framework*, *RelBOSS*.

4 RelBOSS: The Ontology-Based RAAC Framework

Based on our proposed *RAAC Model* (relationship and policy models), we in this section present an ontology-based framework, *Relationship-Based access control Ontology for Software Services (RelBOSS)*. The principal goal of our RelBOSS framework is to realize the concepts of relationship and policy models using a logic-based language. To achieve this goal, we have already identified relevant concepts in the previous section.

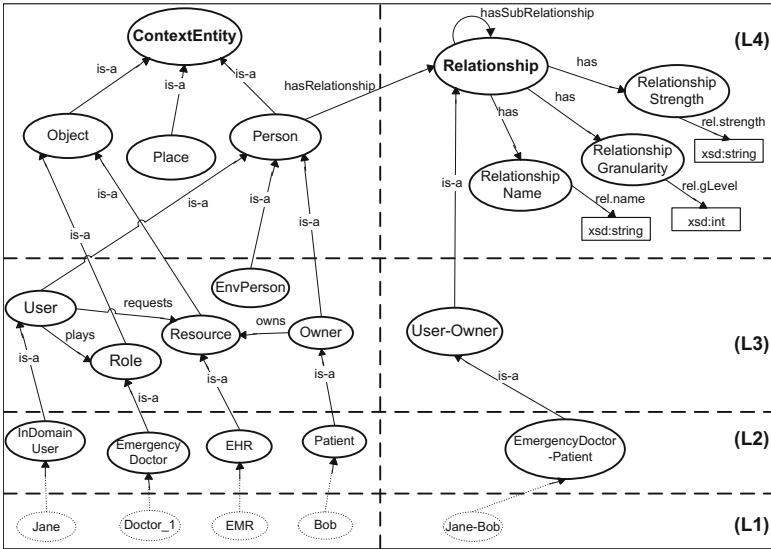


Fig. 1. Core *Relationship* Ontology

4.1 Design Considerations

In the present age, ontology-based modeling technique has been proven as a suitable logical language for modeling dynamic context information (e.g., relationship context information) [4], because of its representational and reasoning viewpoints. We adopt the OWL language [11] as an ontology language to model RelBOSS. The OWL language has been chosen for most ontological applications because of its considered trade-off between computational complexity of reasoning and expressiveness [4]. In order to support the process of inferring implicit knowledge (the *granularity levels* of the relationship, the relationship *strengths*), we define a set of user-defined reasoning rules that are associated with the basic knowledge (currently available context information). Towards this end, the expressivity of OWL is extended by adding SWRL [12] rules to RelBOSS.

4.2 Relationship Ontology

4.2.1 Representation of Relationship - Core Concepts

Figure 1 shows the conceptual view of our *Relationship* ontology (definition of upper concepts/entities and partial definition of domain-specific concepts). The *Relationship* ontology can be divided into four layers. The top layer (*L4*) shows the core relationship-aware concepts for specifying the relationship context information. The middle layer (*L3*) shows the constructs for defining access control-specific core concepts. The middle layer (*L2*) shows the domain-specific concepts that can be modeled using the core concepts. The bottom layer (*L1*) shows the instances based on the healthcare application used in this paper. The constructs in the ontology are summarized as follows.

The layer (*L4*) is structured around a set of core entities, each describing the concepts including *ContextEntity* (for defining access control-specific core

classes/entities: *Person*, *Place* and *Object* types) and *Relationship* (for defining the relationship context information). Each entity is associated with its attributes (represented in *data type* property), relations with other entities (represented in *object* property), and its subclasses (represented in *is-a*, i.e., *is a subclass of* property). An object property *hasRelationship* is used to link the *Person* and the *Relationship* classes. The *Relationship* class is connected with three other classes: *RelationshipName*, *RelationshipGranularity* and *RelationshipStrength*. The *RelationshipName* class contains a data type property named *rel.name* (*xsd:string* type), which denotes the type of the relationship. The *RelationshipGranularity* class contains a data type property named *rel.gLevel* (*xsd:int* type), which denotes the different granularity levels of the relationship existing between the entities participating in a certain situation. The *RelationshipStrength* class contains a data type property named *rel.strength* (*xsd:string* type), which denotes the strengths of the relationship. The *Relationship* class has an object property *hasSubRelationship* to model the relationship hierarchy, so as to achieve the users' service access request at different granularity levels (detail in "Representation of Relationship - Domain-Specific Concepts" Subsection).

The layer (*L3*) defines the access control-specific core entities. The *User*, *Role*, *Resource*, *EnvPerson* and *Owner* classes are the subclasses of the core classes. A built-in property of the OWL, named *is-a*, is used to link between *ContextEntity* class and its subclasses. The relationship between user and owner is represented as *User-Owner* class, which is a subclass of the core class *Relationship*. We also define some of the data type properties: *userIdentity* is the attribute of the class *User*, to capture user identity; *roleIdentity* is the attribute of the class *Role*, to capture role identity; *resourceIdentity* is the attribute of the class *Resource*, to capture requested resource identity; and *connectedPeopleIdentity* is the attribute of the class *Owner*, to capture connected people identity. For the sake of simplicity, Figure 1 does not capture/present these data type properties. Overall, the layers 3 and 4 models the core/general concepts. A general context model specific to access control is proposed in our earlier work [9]. Our relationship ontology extends the existing context model, in order to derive fine-grained relationship (the type/name, granularity levels and strengths of the relationship).

4.2.2 Representation of Relationship - Domain-Specific Concepts

The core Relationship ontology (shown in Figure 1) serves as an entry-point for the domain ontologies. The domain-specific concepts extend the core ontology's corresponding base concepts. It is important for the application developers, providing a way to include domain-specific concepts into the core ontology.

The layer (*L2*) in Figure 1 shows a partial definition of domain-specific entities/concepts for the healthcare domain. For example, the core classes *Role* and *Resource* of the healthcare domain are instantiated into subclasses *Emergency-Doctor* and electronic health records (*EHR*) respectively. The layer (*L1*) shows some example instances based on the healthcare scenario used in this paper. For example, the instances *Jane* and *Bob* are represented as *InDomainUser* and *Patient*, the emergency medical records (*EMR*) is the instance of *EHR*, etc.

The different relationships at various granularity levels of a user's service access request are individually identifiable, so as to achieve fine-grained relationship-aware control over access to services. Towards this end, we consider a detailed ontology with different levels of relationships for the healthcare application. We express the existence of different relationships at fine-grained levels. Figure 2 shows an excerpt of the representation of the *Relationship* ontology for the healthcare domain. In Figure 2, we only model the interpersonal relationships between user and owner (*User-Owner* class). As such, the *Relationship* class contains an important property (*xsd:int* type) named *rel.gLevel*, which indicates the relationship granularity levels. For instance, a medical practitioner can access a patient's electronic health records with the "*Doctor-Patient*" relationship (*rel.gLevel = 0*). However, he/she can access the emergency medical records, if he/she is an emergency doctor of the patient (*rel.gLevel = 1*, i.e., "*EmergencyDoctor-Patient*" relationship).

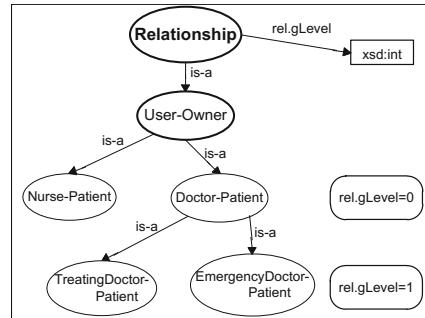


Fig. 2. An Excerpt of Relationship Ontology for Healthcare

In Figure 2, we only model the interpersonal relationships between user and owner (*User-Owner* class). As such, the *Relationship* class contains an important property (*xsd:int* type) named *rel.gLevel*, which indicates the relationship granularity levels. For instance, a medical practitioner can access a patient's electronic health records with the "*Doctor-Patient*" relationship (*rel.gLevel = 0*). However, he/she can access the emergency medical records, if he/she is an emergency doctor of the patient (*rel.gLevel = 1*, i.e., "*EmergencyDoctor-Patient*" relationship).

4.3 Reasoning About Relationship

A more flexible reasoning mechanism is user-defined reasoning. Through the creation of SWRL user-defined reasoning rules, the relationship context information can be deduced from the basic context information. A set of reasoning rules are specified for implicit knowledge reasoning, which reasons about the implicit knowledge (the *name* or *type* of the relationship, and the *granularity levels* and *strengths* of that relationship) convoyed by the specification.

4.3.1 Example Rules

We specify a set of SWRL reasoning rules, to derive the name of the relationship between User and Owner. For example, Rule #1 in Table 2 is used to derive the name (*rel.name*) of the relationship (which is "*User-Owner*"). The rule states that *if* a User requests a Resource which is owned by a Patient (Owner), *then* the relationship name/type is "*User-Owner*".

Another set of SWRL reasoning rules are also specified in Table 2, to derive the granularity levels (*rel.gLevel*) of the relevant relationship. For example, *if* the relationship between user and owner is "*EmergencyDoctor-Patient*", *then* the granularity level is 1 (see Rule #2 in Table 2, where a user plays the Emergency Doctor (ED) role). *If* the user in a Doctor role, *then* the granularity level is 0 ("*Doctor-Patient*").

We further specify a set of reasoning rules to derive the *strengths* of the relationship, based on the basic context information. An example rule shown in Table 2 (see Rule #3) identifies the relationship strengths (*rel.strength*) is "*Strong*". Note that this rule has used the basic information, the user's profile information

Table 2. User-defined SWRL Reasoning Rules

Rule	Relationship	Reasoning Rules
1	Relationship Name	$\text{User}(\text{?u}) \wedge \text{Resource}(\text{?res}) \wedge \text{requests}(\text{?u}, \text{?res}) \wedge \text{swrlb:equal}(\text{?res}, \text{"EMR"}) \wedge \text{Owner}(\text{?o}) \wedge \text{Resource}(\text{?res}) \wedge \text{owns}(\text{?o}, \text{?res}) \wedge \text{Relationship}(\text{?rel}_1) \wedge \text{hasRelationship}(\text{?u}, \text{?rel}_1) \wedge \text{hasRelationship}(\text{?o}, \text{?rel}_1) \wedge \text{RelationshipName}(\text{?rN}) \wedge \text{has}(\text{?rel}_1, \text{rN}) \rightarrow \text{rel.name}(\text{?rN}, \text{"User-Owner"})$
2	Granularity Level	$\text{User}(\text{?u}) \wedge \text{Role}(\text{?r}) \wedge \text{plays}(\text{?u}, \text{?r}) \wedge \text{swrlb:equal}(\text{?r}, \text{"ED"}) \wedge \text{Relationship}(\text{?rel}_1) \wedge \text{RelationshipGranularity}(\text{?rG}) \wedge \text{has}(\text{?rel}_1, \text{rG}) \rightarrow \text{rel.gLevel}(\text{?rG}, 1)$
3	strengths	$\text{User}(\text{?u}) \wedge \text{Owner}(\text{?o}) \wedge \text{RelationshipStrength}(\text{?rS}) \wedge \text{userIdentity}(\text{?u}, \text{?uID}) \wedge \text{connectedPeopleIdentity}(\text{?o}, \text{?cpID}) \wedge \text{swrlb:equal}(\text{?uID}, \text{?cpID}) \wedge \text{has}(\text{?rel}_1, \text{rS}) \rightarrow \text{rel.strength}(\text{?rS}, \text{"Strong")}$

(*userIdentity*) and the patient's profile information (*connectedPeopleIdentity*). This basic information can be obtained through a program (e.g., in Java) from the data sources (e.g., RDBMS). For the motivating scenario, this rule determines that Jane and Bob has a relationship with "Strong" strength, because Jane is a treating doctor of patient Bob. *If* Jane is not a treating doctor of patient Bob, *then* the relationship strengths is "Weak".

4.4 The RAAC Policy Ontology

In the following, we present the two main parts of our RAAC policy ontology: *policy specification* (integrating relationship context information into the access control policies) and *policy enforcement* (determining access control decisions).

4.4.1 Policy Specification

Figure 3 shows the top-level conceptual view of our RAAC policy ontology.

We model our policy ontology based on the 3Ws: *who* (user/role) wants to access *what* (service) and *when* (relationship context information). A *RAAC Policy* is a collection of entities and it includes *User*, *Role*, *Relationship*, *Service*, and *AccessDecision* classes. The policy model allows the specification of RAAC policies for software services, where the authorized users (an authorized user is a user with a role instance) can invoke appropriate services in terms of the relevant relationship context information. It uses the *Relationship* concept from the relationship ontology (presented in Figure 1), to capture the relationship context information. The RAAC policy ontology also uses the concepts, *User*, *Role*, and *Resource* from the relationship ontology in order to capture the relevant context information at runtime. The reused classes are shown in shaded ellipses (see Figure 3).

We characterize a *Service* as a composition of *Operations* on *Resources*, through which information resources or software services are invoked by the users or service requesters. An object property, named *composedOf*, is used to link between *Service* and *Resource* classes, and *Service* and *Operation* classes (see Figure 3). Table 3

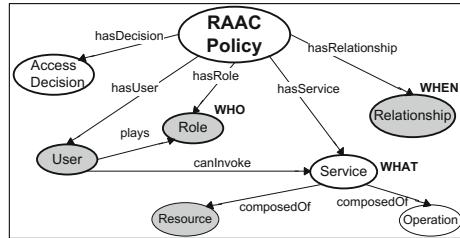
**Fig. 3.** The RAAC Policy Ontology

Table 3. *Service Definition* and An Example Service *Service_EMR_write*

<pre> <owl:ObjectProperty rdf:ID="composedOf"> <rdfs:domain rdf:resource="#Service"/> <rdfs:range> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#Resource"/> <owl:Class rdf:about="#Operation"/> </owl:unionOf> </owl:Class> </rdfs:range> </owl:ObjectProperty> </pre>
<pre> <Service rdf:ID="Service_EMR_write"> <composedOf rdf:resource="#Resource_EMR"/> <composedOf rdf:resource="#Operation_write"/> </Service> ... </pre>

shows a service definition in OWL, a *Service* is composed of the collection/union of *Resource* and *Operation* (see the top part of the Table 3). An example of a service, named *Service_EMR_write* is shown in Table 3 also, which is composed of the *write* operation on a resource (e.g., a patient's emergency medical records (*EMR*)) (see the bottom part of the Table 3).

Definition 3 (RAAC Policy Specification). A relationship-aware access control policy captures the *who/what/when* dimensions which can be read as follows: a *RAACP*olicy specifies that a *User* who is playing a *Role* has *AccessDecision* ("Granted" or "Denied") to *Service* if the relevant *Relationship context information* is satisfied.

Table 4. An Example RAAC Policy (Upper Level)

<pre> <RAACP> <hasUser rdf:resource="#User_plays_ED"/> <hasRole rdf:resource="#Role_EmergencyDoctor_ED"/> <hasService rdf:resource="#Service_EMR_write"/> <hasRelationship rdf:resource="#Relationship_name_gLevel_strength"/> <hasDecision rdf:resource="#AccessDecision_Granted"/> </RAACP> ... </pre>
--

Table 4 shows the policy (which is mentioned in Example 4) written in OWL that is related to our application scenario. In this policy, the access decision ("Granted" decision) is based on the following policy constraints: *who* the user is (user's *role*, e.g., "*ED*"), *what* service being requested (e.g., "*writeEMR()*"), and *when* the user sends the request (*relationship context information*). The '*rel.name*', '*rel.gLevel*', and '*rel.strength*' regarding this *Relationship* are derived based on the SWRL reasoning rules (shown in Table 2).

4.4.2 Policy Enforcement/Evaluation

The specified RAAC policies are enforced to ensure the appropriate use of information resources or software services. Towards this end, we have implemented the *RAAC Policy Enforcement Point* and *RAAC Policy Decision Point* in Java, inspired by the XACML architecture [13]. The prototype architecture is presented in the next Section (see Figure 4). Using our developed prototype, the policy enforcement (relationship-aware access control decisions) is performed according to the following data flow sequences:

1. The users (service requesters) send the access requests to access the resources/services which get processed by our developed *RAAC Policy Enforcement Point*.
2. Once receiving the request for service access, the *RAAC Policy Enforcement Point* queries the applicable access control policies and the relevant relationship context information to the *RAAC Policy Decision Point*.
3. Based on the selected access control policies in the policy ontology, the *RAAC Policy Decision Point* makes the access control decision by runtime assessing the relationship context information from the relationship ontology.
4. The decision made by *RAAC Policy Decision Point* is passed to *RAAC Policy Enforcement Point*. If it is admissible, the access request is granted, otherwise, it is denied. Being granted to access the services (i.e., if the decision is “granted”) means that the users are able to access the services. If the decision is “denied”, a denied response is sent back to the users.

During the policy evaluation phase, an *access query* is used to check the user’s access request (*who can access what services*). We use the query language SQWRL [14], which is based on OWL and SWRL. When a service access request comes, the *RAAC Policy Decision Point* identifies the applicable policies using the policy ontology and identifies the relationship context information using the relationship ontology. Then the formulated access query is used to check the user’s request to access the services using both the policy constraints and the relevant relationship context information. For the application example in the motivating scenario, the defined access query (see an access query in Table 5) is used to determine whether the access is *Granted* or *Denied*. One of the entries in the access query results satisfies Jane’s service access request (“*EMR_write*”) is *Granted* (see Table 6).

Table 5. An Example Access Query (Simplified)

RAACPoliy(?policy) \wedge User(?user) \wedge Role(?role) \wedge rel.name(?rel_nM) \wedge
rel.gLevel(?rel_gL) \wedge rel.strength(?rel_sT) \wedge Service(?service) \wedge AccessDecision(?decision)
\rightarrow sqwrl:select(?user, ?role, ?rel_nM, ?rel_gL, ?rel_sT, ?service, ?decision)

Table 6. Access Query Result (Shown Only One Entry)

?user	?role	?rel_nM	?rel_gL	?rel_sT	?service	?decision
Jane	ED	User-Owner	1	Strong	EMR_write	Granted

5 Prototype Implementation and Evaluation

5.1 Prototype Implementation

Figure 4 shows a high-level conceptual view of our prototype framework in Java2 SE using widely supported tools.

We have used the Protégé-OWL API [15] to implement the relationship and policy ontologies (RelBOSS ontology KB). Once we expressed the set of policies that apply to a system as a knowledge base, run time reasoning can be

performed for access control decision. During the policy evaluation phase, an access query is used to process the user's service access request. We have used the SWRL rules to evaluate the policies. We have used the Jess Rule Engine [16] for executing the SWRL rules. In particular, the query language SQWRL [14], which is based on OWL and SWRL, is adopted to process service access requests. Due to space limitation, the complete description of the prototype can not be included in this paper.

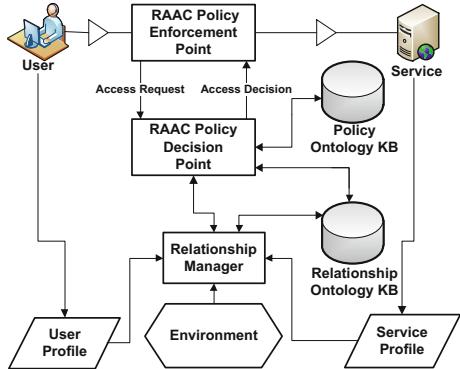


Fig. 4. Our Prototype Architecture

5.2 Developing a RAAC Application for Healthcare

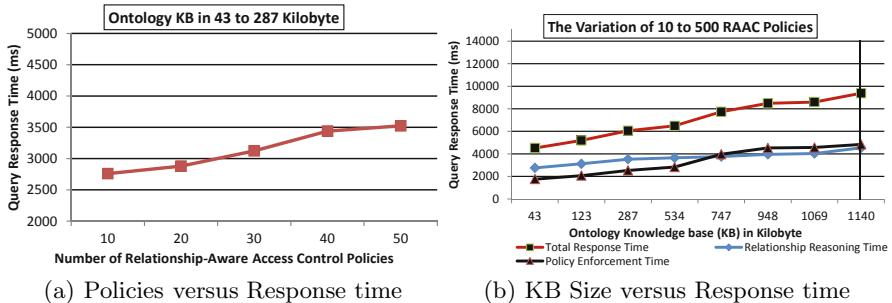
Using the prototype, we develop a relationship-aware access control application in the domain of electronic health records management system, and provide a healthcare case study (i.e., test-scenario based implementation). The main goal that we aim with this application is to access different medical records of patients based on the relevant relationship context information.

Case Study. Consider our application scenario, where Jane, by playing an emergency doctor (ED) role, wants to access the requested service *writeEMR()* (i.e., the write access to the emergency medical records (EMR) of patient Bob). Our *Relationship Ontology* captures the relevant relationship context information based on the currently available information and relationship specification rules. Our *Policy Ontology* captures the relevant RAAC policy. Based on this information, the framework returns the RAAC decision, i.e., Jane's service access request is **Granted** (see an access query in Table 5 and result in Table 6), because the ontology captures relevant relationship context information, and satisfies a RAAC policy which is stored in the policy base (ontology KB).

5.3 Experimentation

With the goal of evaluating the runtime performance of the various stages of our prototype framework, we have conducted a range of experiments on a Windows XP Professional operating system running on Intel(R) Core(TM) Duo T2450 @ 2.00 GHz processor with 1GB of memory.

Measurement. We have measured the *end-to-end query response time* (T) of the different aspects of *access request processing*, i.e., the time taken from the arrival of the user's service access request to the end of its execution (the sum of (i) *relationship context reasoning time* and (ii) *RAAC policy enforcement time*). The first measure indicates how long it took to infer the relationship context



(a) Policies versus Response time

(b) KB Size versus Response time

Fig. 5. Query Response Time Over Different Numbers/Size of Policies/KB

information (by capturing the currently available context information). The second measure indicates how long it took to process a user's service access request (by incorporating the inferred relationship context into the access control process and making relationship-aware access control decision). We separate the time delay for loading the relationship ontology and policy ontology from the access request processing time. The ontology loading occurs when the system runs the first time. In general, the most important parameter here is *the access request processing time*, which is of more interest or significance than the ontology loading time.

Experimental Results and Analysis. In the first test, we have evaluated the query response time when the number of access control policies increased. First, we have selected 10 policies with respect to 10 different health professional roles: *General Practitioner, Emergency Doctor, Registered Nurse*, etc [17]. We have varied the number of policies up to 50 with 50 health professional roles. Each of these variations is executed 10 times. The average value of the 10 execution runs is used for the analysis (see Figure 5(a)). The test results show that the query response time (T) increases when the number of policies increases. It varies between 2.7 and 3.5 seconds for the variation of 10 to 50 relationship-aware access control (RAAC) policies. In Figure 5(a), we can see that the query response time seems to be linear. Overall, the system performance is acceptable.

In the second test, we have again evaluated the response time (relationship reasoning time and policy enforcement time) over various size of the ontology knowledge base (KB). We have varied the number of policies up to 500 with respect to 138 different health professionals [17] (i.e., 138 roles). To measure the query response time, we have run the experiment 10 times. The results in Figure 5(b) present the average response time over various size of ontology KB. The computational overhead increases at a linear rate (note that the scales are not linear in the horizontal and vertical axes). At the point of the KB being 1140 kilobytes (where we have specified 500 policies for the 138 health professional roles), it takes approximately 9 seconds to process the access request.

In the last test, we have focused on measuring the ontology loading time (the relationship ontology and policy ontology) of our prototype with the increasing size of KB. The test results in Figure 6 show that the ontology loading time

in increasing the size of the KB varies from 9 to 21 seconds approximately. Similar to previous test, in Figure 6, the scales are not linear in the horizontal and vertical axes. This results do not have impact on the end-to-end query response time (T) as ontology loading is only done at the start of the system.

In general, the query response time increases with the increasing number of access control policies (over various size of the ontology KB). Although, this seems a little expensive, at the point where we have modeled 500 relationship-aware access control policies for the 138 health professional roles (see Figure 5(b)), the performance can be improved by using more powerful machine.

6 Related Work and Discussion

Considering research in the field of relationship information, in this section we briefly highlight several existing research works. Some works integrate the *person-centric* relationship information and other works incorporate the *location-centric* relationship information in the access control processes.

Friend of a Friend (FOAF) [18] is one of the most used semantic web ontologies, which describes persons, using a set of properties that provide information about a person such as *name*, *gender*, *email*, and *relationship* with other persons. While FOAF considers only the generic relationships between two people, corresponding to the ones modeled by the *foaf:knows* property, more specialized interpersonal relationships describing the interconnection between two people at a fine-grained level (e.g., the relationship between the medical practitioner and patient is a “EmergencyDoctor-Patient” with “strong” strength) is an important aspect for relationship-aware access control applications. Some other research works have attempted to adopt and extend the FOAF ontology for deriving social relationships based upon the user’s daily communication with other people [19][20]. They extend the *foaf:knows* property with sub-properties such as *NeighborOf* and *ColleagueOf*. Same as basic FOAF model, these approaches do not allow the specification of different relationship context information such as the different granularity levels and strengths associated with a relationship. On the other hand, Toninelli et al [21] adopt the FOAF ontology, representing the types of resources and the relationships between them, in order to develop their Yarta middleware. They only consider the type of the relationships (e.g., *isMember(Agent, Group)*). A major difference of our relationship model with respect to these FOAF and FOAF-extended models is that, we not only consider the *person-centric* relationship (relationship type) but also consider the different *granularity levels* and *strengths* of that interpersonal relationship based on the currently available context information.

Some recent access control research works (e.g., [1], [2], [7], [8]) have integrated the *person-centric* relationship information into policy specifications, each

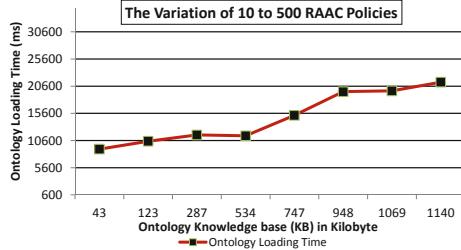


Fig. 6. KB Size versus Ontology Loading Time

of them having different origins, pursuing different goals and often, by nature, being highly domain-specific. Carminati et al [1] have proposed an access control approach, which allows the specification of access control policies for online resources in order to model various aspects of online social networks. They propose access control policies that depend on the relationship between users (e.g., *friend*, *friend-of-friend*), and relationships between user and resource. Zhang et al [7] have proposed an approach named relation-based access control (ReBAC), in which the access permissions are formalized as binary relations between subjects and objects. They consider subject, object and permission as the compulsory access control components. Fong et al [8] have presented a relationship-based access control approach named ReBAC. They consider the relationships between individual users (e.g., *professional association*) in the access control policies. Squicciarini et al [2] have proposed a privacy protection mechanism for social networks named PriMa, which specifies access control policies for users profile information. PriMa considers the relationship between resource owner and requester (e.g., *common friend*) in the access control processes.

These relationship-aware approaches consider the expression of access control policies in terms of the *person-centric* or interpersonal relationships to control access to information resources or software services. However, these approaches do not address and model the *different granularity levels* and *strengths* of the interpersonal relationship. Furthermore, these existing approaches are domain-specific and they are hard to be implemented for various access control applications in today's dynamic context-aware environments. Different from these approaches, we, therefore, introduce in this paper a *Relationship Ontology*, to dynamically identify much richer forms of relationships among relevant entities with different granularity levels and strengths of that relationship, and a *RAAC Policy Ontology*, to provide relationship-aware access to information resources based on the inferred relationship context information.

Several other access control research works (e.g., [22], [23]) have incorporated the *location-centric* relationship information into policy specifications. Ardagna et al [22] have proposed an access control approach for location-based services. They integrate three types of location-based conditions into their policy model, position-based, movement-based and interaction-based conditions. Tarameshloo and Fong [23] have proposed an access control approach for Geo-social computing systems, which allows users to declare their current locations and uses these declared locations to make access control decisions. They propose access control policies that depend on the location declarations/claims by users, e.g., *nearby* users can allow access. These access control approaches only consider the *location-centric* relationships for making access control decisions. However, these approaches do not address and model the *person-centric* relationships.

7 Conclusion and Future Work

Dynamic and context-aware environments require new access control frameworks, changing the focus of access control paradigm from identity/role-based to relationship-based. Existing relationship-aware access control (RAAC) approaches integrate the relationship dimension into policy specifications. How-

ever, these approaches do not provide adequate supports for identifying the different granularity levels and strengths of the relationship. One of the main contributions of this paper is the formal *RAAC Model* for specifying the relevant relationship information with different granularity levels and strengths, and the corresponding relationship-aware access control policies. Another contribution of this paper is an ontology-based framework, *RelBOSS*, in order to realize RAAC model using OWL and SWRL. The RelBOSS framework defines and enforces the relationship-aware access control policies by incorporating the relevant relationship context information. We have demonstrated the practical applicability of our framework through the implementation of a software prototype in the healthcare domain and presented a case study. The case study shows that our framework is able to capture relevant relationship context information at runtime and invoke software services in a relationship-aware manner. In addition, the experimental results show the feasibility of our framework.

Our framework can be extended to model and capture other types of relationships (e.g., a fine-grained “location-centric” relationship can be identified by using the spatial relations between different entities, i.e., *nearby* or *co-located* persons). We plan to apply our framework/system to the real world and further examine performance and optimisation issues as future work.

Acknowledgment. Jun Han is partly supported by the Qatar National Research Fund (QNRF) under Grant No. NPRP 09-069-1-009. The statements made herein are solely the responsibility of the authors.

References

1. Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M., Thuraisingham, B.: A semantic web based framework for social network access control. In: SACMAT, pp. 177–186 (2009)
2. Squicciarini, A., Paci, F., Sundareswaran, S.: Prima: an effective privacy protection mechanism for social networks. In: ASIACCS, pp. 320–323 (2010)
3. Weiser, M.: Some computer science issues in ubiquitous computing. Commun. ACM 36(7), 75–84 (1993)
4. 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, 161–180 (2010)
5. Dey, A.K.: Understanding and using context. Personal Ubiquitous Computing 5(1), 4–7 (2001)
6. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: PerCom Workshops, pp. 18–22 (2004)
7. Zhang, R., Giunchiglia, F., Crispo, B., Song, L.: Relation-based access control: An access control model for context-aware computing environment. Wirel. Pers. Commun. 55(1), 5–17 (2010)
8. Fong, P.W., Siahaan, I.: Relationship-based access control policies and their policy languages. In: SACMAT, pp. 51–60 (2011)
9. Kayes, A.S.M., Han, J., Colman, A.: An ontology-based approach to context-aware access control for software services. In: Lin, X., Manolopoulos, Y., Srivastava, D., Huang, G. (eds.) WISE 2013, Part I. LNCS, vol. 8180, pp. 410–420. Springer, Heidelberg (2013)

10. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. *IEEE Computer* 29, 38–47 (1996)
11. OWL (February 2014), <http://www.w3.org/2007/owl/>
12. SWRL (February 2014), <http://www.w3.org/submission/swrl/>
13. Moses, T.: Extensible access control markup language (xacml). In: OASIS Standard (2005), <https://www.oasis-open.org/committees/xacml/>
14. O'Connor, M.J., Das, A.K.: Sqwrl: A query language for owl. In: OWLED (2009)
15. Protégé-OWL (February 2014), <http://protege.stanford.edu/>
16. JESS: Jess rule engine (February 2014), <http://herzberg.ca.sandia.gov/>
17. ASCO: Australian standard classification of occupations, health professionals (February 2014), <http://www.abs.gov.au/>
18. FOAF: Friend of a friend, <http://xmlns.com/foaf/spec/20100809.html> (February 2014)
19. RELATIONSHIP: A vocabulary for describing relationships between people (February 2014), <http://vocab.org/relationship/>
20. Devlic, A., Reichle, R., Wagner, M., Pinheiro, M.K., Vanrompay, Y., Berbers, Y., Valla, M.: Context inference of users' social relationships and distributed policy management. In: PerCom Workshops, pp. 1–8 (2009)
21. Toninelli, A., Pathak, A., Issarny, V.: Yarta: A middleware for managing mobile social ecosystems. In: GPC, pp. 209–220 (2011)
22. Ardagna, C.A., Cremonini, M., di Vimercati, S.D.C., Samarati, P.: Access control in location-based services. In: Privacy in Location-Based Applications, pp. 106–126 (2009)
23. Tarameshloo, E., Fong, P.W.L.: Access control models for geo-social computing systems. In: SACMAT, pp. 115–126 (2014)

A QoS-Aware, Trust-Based Aggregation Model for Grid Federations

Antonello Comi¹, Lidia Fotia¹, Fabrizio Messina², Domenico Rosaci¹,
and Giuseppe M.L. Sarnè³

¹ DIIES, University “Mediterranea” of Reggio Calabria,
Via Graziella, Feo di Vito 89122 Reggio Calabria (RC), Italy
`{antonello.comi,lidia.fotia,domenico.rosaci}@unirc.it`

² Dipartimento di Matematica e Informatica, University of Catania,
V.le Andrea Doria 6, 95125 Catania, Italy
`messina@dm.unict.it`

³ DICEAM, University “Mediterranea” of Reggio Calabria,
Via Graziella, Feo di Vito 89122 Reggio Calabria (RC), Italy
`sarne@unirc.it`

Abstract. In this work we deal with the issue of optimizing the global Quality of Service (QoS) of a Grid Federation by means of an aggregation model specifically designed for intelligent agents assisting Grid nodes. The proposed model relies on an algorithm, called FGF (Friendship and Group Formation), by which the nodes select their partners with the aim of maximizing the QoS they perceive when a computational task requires the collaboration of several Grid nodes. In the proposed solution, in order to assist the selection of the partners, a suitable trust model has been designed. Since jobs sent to Grid Federations hold complex requirements involving well defined resource sets, trust values are calculated for specific sets of resources. We also provide a theoretical foundation and some experiments to prove that, by means of the adoption of the FGF algorithm suitably supported by the proposed trust model, the Grid Capital (which reflect the global QoS) of the Grid Federation is eventually improved.

1 Introduction

Grid Computing [16] has recently evolved to the Federated Grids architecture [25,38], in which Grid brokers [1,16] and Grid institutions can share resources among different types of Grid infrastructures, resulting as a more flexible approach than that derived by the classical and well known concept of Virtual Organization (VO).

The development of Federated Grids has been strongly encouraged by the increasing complexity of Grid tasks [5] submitted by companies and institutions [45] and supported by the quick and recent advancements of Virtualization technologies [4,26,41]. In particular, such improvements have introduced more flexibility in managing resources, software deployment activities, configuration of

software platforms, and so on. As a consequence, Grid infrastructures are characterized by a high dynamic, such that companies and institutions are allowed to join or leave different Grid infrastructures in a very easy way, reconfigure their own nodes and resources, modify their own role into the VO, their decisional processes and so on.

In the above scenario, Grid users are encouraged to send job requests characterized by more and more complex requirements, e.g. jobs might require the execution of “inter-grid” workflows, composite services, etc. Consequently, on this base, federated Grid brokers have to deal with complicated tasks of resource allocation by evaluating requirements involving a huge set of *federated resources* in order to satisfy job requirements [8]. This is mainly due to the fact that Grid Federations are intended to exploit the potential *collaboration* between Grid VOs which are able to provide highly specialized, and not trivial, contributions to the result of the expected computation.

Furthermore, federated resources also need to be allocated by paying particular attention to use them in an efficient fashion. For instance, complex job requirements must be satisfied first, and at the same time, choices that might cause unbalanced resources allocations or poor performances should be avoided as much as this is possible. It is even conceivable that the establishment of Grid Federations brings the participants to join with a highly “competitive” scenario. Although competition usually leads better performances, it enhances the presence of possible malicious behaviors. Trust-based systems can help to solve and/or mitigate this problem and, therefore, it is important that competitive Grid Federations are supported by a well suited trust model [7,10,23] in order to quantify the level of performance and reciprocal trust of Grid nodes.

Basing on the considerations above, in this work we propose an approach aiming to maximize the measured “performance” or, in other words, the global Quality of Service (QoS) perceived within the Grid Federation. In this approach we focus on the concept of *resource sets*, i.e. the sets of computational resources characterizing complex jobs in Grid Federations. The proposed solution is based on the use of software agents [17], which, in our approach, manage every node of the Grid Federation. Moreover, the concept of *agent aggregation* (i.e. groups and friendship) is exploited as the basis of collaboration between federated nodes. By combining the use of a Trust model, which provides to compute some measures of reliability and reputation, in turn combined in a unique synthetic trust measure, we propose an algorithm for agent Friendship and Group Formation (FGF) in order to maximize the “global utility”, i.e. the *Grid Capital* of the whole Grid federation. Finally, we prove that the execution of the FGF algorithm, which is supported by the dissemination of trust information, allows Grid federated Brokers/Nodes to improve both individual and global satisfaction.

The plan of the paper is as follows. In Section 2 we describe a reference grid scenario for competitive agents. Then, Section 3 introduces our trust model. Section 4 presents the FGF algorithm for forming friendships and groups, while in Section 5 we provide some theoretical results which prove the validity of the proposed FGF algorithm. In Section 6, some experiments devoted to verify

the effectiveness of the FGF algorithm are presented. Finally, in Section 7 we discuss some related work and in Section 8 draw our conclusions and introduce our ongoing researches.

2 The Competitive Grid Scenario

This work is based on a generic scenario in which a set N of Grid Nodes provide Grid Services within a Grid Federation [44] to a set C of clients (Grid Users).

Grid Services on Competitive Grids. A Grid Service generally involves the execution of workflows [46] and/or the provisioning of composite services [36], therefore a certain amount of different resources is generally required for their execution, therefore, we assume to classify each Grid Service on the basis of such required resources. More in detail, we model the set of heterogeneous resources which can be found on N as a finite number of R incremental sets of resources, where the last set (i.e. the R -th) includes all the resources belonging to N . Moreover, the Grid scenario we take into consideration is “competitive” and we assume that when the generic client $c_j \in C$ benefits for the service s^r (with $1 \leq r \leq R$) provided by n_i , it will have to pay a fixed price p to n_i based on the set r of consumed resources.

Agents, skills and feedbacks. We also assume that each Grid node is assisted by a software agent [17], supporting it in providing services to Grid Users. Let A be the set of these agents, where the agent $a_i \in A$ is associated with the node $n_i \in N$. Furthermore, each agent a_i is characterized by a “skill” mapping, $\sigma_i(r) \in [0, 1] \subseteq \mathbb{R}$, where 1/0 means the maximum/minimum quality in providing a service requiring that specific set of resources (i.e. r). When the client c_j benefits from the service s^r provided by the agent a_i (i.e. node n_i), it returns a feedback f to a_i , where f is a real value belonging to $[0..1]$, such that 1 means the maximum satisfaction perceived by c_j for s^r and, vice versa, 0 means the minimum perceived satisfaction.

Agent aggregation. In the presented model we rely on the concepts of agent friendships and groups, by supposing that each agent a_i maintains a set F_i of friend agents, such that $F_i \subseteq A$, and a set of groups $G_i = \{g_{i_1}, \dots, g_{i_k}\}$ where $\bigcup_{1 < l < k} g_{i_l} \subseteq A$.

Agent collaboration for service provisioning. To satisfy the request for a service s^r , the agent a_i can require the support of another node n_j (i.e. a_j) that in turn can accept or refuse it¹. If a_j accepts and it is a friend of a_i or it is in the same group with a_i , this contribution will be provided for free; otherwise, a_j will require the payment of a fee p_s to the agent a_i after the service has been consumed. In order to select the best agents which can collaborate for the service s^r , the agent a_i can ask the opinion of other agents. In particular, a_i

¹ We assume that each agent can perform at most X support requests.

can send a request to an agent a_k to obtain a recommendation $rec_j(r)$ about the skill $\sigma_j(r)$ of a_j for a given service s^r . In turn a_k can accept or refuse the request for $rec_j(r)$ ². If a_k accepts and it is a friend of a_i or it is in the same group with a_i , this recommendation will be provided for free; otherwise, after the recommendation has been provided by a_k , a price p_r should be paid by a_i to a_k . The final choice is performed by the agent a_i based on the Trust model described into Section 3, which takes into account also the *reliability* of the node a_j . It is updated by taking into account the feedback provided by the Grid User who consumed the service.

Looking for agents and groups. In such a scenario, the names of agents, groups and agents belonging to each group have to be appropriately registered, such that their names and locations can be easily retrieved. For this aim we assume that those entities are registered by means of a service named Directory Facilitator (*DF*), which in turn is published by the various Grid VO (Virtual Organizations) by means of the service infrastructure GIS (Grid Information Services) [11].

3 The Trust Model

In the described grid context, the presence of competitive agents compels us to also consider their possible misbehaviors. For instance, an agent could receive from another agent (*i*) less resources from those corresponding to its actual skill coefficient σ or (*ii*) an unfairly recommendation about the skills of a third agent. To manage such misbehaviors, in multi-agent environments a common solution is represented by trust systems [24]. To this purpose, in the proposed model, for each agent a_j to which it interacted in the past, each agent a_i maintains a triple of values (α, β, γ) , respectively called *Reliability*, *Honesty* and *Reputation* that belong to $[0, 1] \in \mathbb{R}$.

Reliability ($\alpha_{ij}(r)$). The first value, denoted by $\alpha_{ij}(r)$, is the *reliability* of a_j in providing a set r of resources, and represents how much a_i trusts a_j in its capability to provide resources for a service s^r . More in detail, for each feedback f received by a_i for a service s^r , if a_i was supported in its task by a_j , the feedback f includes a contribution f_j^* due to a_j , that will be assigned to a_j . Obviously, if a_i completely delegated a_j to provide s^r , then it will be $f_j = f = f_j^*$. Finally, the reliability is computed as the arithmetical mean of all the feedback received by a_i for all the services which required the set of resources r and the collaboration of a_j :

$$\alpha_{i,j}(r) = \begin{cases} \frac{1}{l} \sum_{m=1}^l f_j^m & \text{if } l \neq 0 \\ \bar{\alpha} & \text{if } l = 0 \end{cases} \quad (1)$$

² We assume that each agent can perform at most Y recommendation requests.

Honesty ($\beta_{i,j}(r)$). The second value, denoted by $\beta_{i,j}(r)$, is referred as the honesty of a_j in giving a recommendation to a_i about the capability of another agent to provide a set r of resources. It is computed by comparing the feedbacks f_{x_1}, \dots, f_{x_l} obtained by a_i for some agents a_{x_k} suggested by a_j . In other words, it is the arithmetical mean of all the difference between each f_k and the associated recommendation rec_j^l provided by a_j about some agents a_{x_l} . More formally:

$$\beta_{i,j}(r) = \begin{cases} \frac{1}{l} \sum_{m=1}^l |rec_j^m(r) - f_{x_m}(r)| & \text{if } l \neq 0 \\ \overline{\beta} & \text{if } l = 0 \end{cases} \quad (2)$$

Reputation ($\gamma_{ij}(r)$). The last value is the *reputation* of a_j with respect to the set of resources r , denoted by $\gamma_j(r)$, representing how much the agents interrogated by a_i estimated the capability of a_j about the resource set r provided by a_j . The reputation $\gamma_{ij}(r)$ is computed as the mean of all the recommendations that each other agent provided to a_i about a_j on r weighted by its *honesty* value. More formally:

$$\gamma_{i,j}(r) = \begin{cases} \frac{1}{l} \sum_{m=1}^l rec_j^m(r) \beta_{im}(r) & \text{if } l \neq 0 \\ \overline{\gamma} & \text{if } l = 0 \end{cases} \quad (3)$$

Note that in the equations described above, the values $\overline{\alpha}$, $\overline{\beta}$ and $\overline{\gamma}$, called “cold start” values, are used in the case any interaction with other agents occurred in the past for a_i with respect to the considered set of resources r as, for instance, for the new coming agents.

Finally, it is possible to compute the synthetic measure of trust of a_i about an agent a_j with respect to the set of resources r , denoted by $\tau_{i,j}(r)$, as:

$$\tau_{i,j}(r) = \delta_i \cdot \alpha_{i,j}(r) + (1 - \delta_i) \cdot \gamma_j(r)$$

where $\delta \in [0, 1] \subset \mathbb{R}$ is used to weight the relevance assigned by a_i to the reliability with respect to the reputation.

4 The Friendship and Group Formation (FGF) Algorithm

As described in Section 2, when an agent a_i asks for a contribution or a recommendation to another agent a_j which is a friend or a member of its same groups, it will be for free. The difference is that in the first case a_i and a_j mutually accepted to become friends in the past, while in the latter a_j could be only a group mate and not also a friend for a_i .

In the above context, a_i can build two sets of preferred agents for each set of resources r , namely:

- a set PC_i^r storing the *preferred contributors* agents which a_i interacted in the past for a contribution falling in r and having (i) the X highest trust values $\tau(r)$ and (ii) a trust value greater than the threshold τ^{min} .

- a set PR_i^r storing the *preferred recommenders* agents which a_i interacted in the past for a suggestion falling in r and having (i) the Y highest honesty values $\beta(r)$ and (ii) a honesty value greater than the threshold β^{min} .

Basing on the definition of trust and honesty provided in Section 3, in order to maximize the performance of the services provided by a_i in collaboration with other agents, its own sets F_i (friends) and $g \in G_i$ (groups) should only include the agents belonging to PC_i^r and PR_i^r for all the set of resources:

$$\bigcup_{r \in R} (PC_i^r \cup PR_i^r) = F_i \bigcup_{g \in G_i} g \quad (4)$$

In order to adopt a convenient notation, we will use the following definitions:

$$PA_i^r = PC_i^r \cup PR_i^r; \quad AG_i = \bigcup_{g \in G_i} g$$

and therefore Equation 4 can be written as

$$\bigcup_{r \in R} PA_i^r = F_i \bigcup AG_i \quad (5)$$

In particular, when Equation 5 is not verified, we have to take into account the two cases described below.

Loss of performance (L): $(\bigcup_{r \in R} PA_i^r) - (F_i \bigcup AG_i) \neq \emptyset$, i.e. there are some agents belonging to the set $\bigcup_{r \in R} PA_i^r$ but not to the set $F_i \bigcup AG_i$. The consequence is represented by the possible *loss of performance*, in providing Grid Services, due to the selection of one of these agents. The issue above can be measured by calculating, for the set $\bigcup_{r \in R} PC_i^r$ (resp. $\bigcup_{r \in R} PR_i^r$), the difference $|\tau_{i,j}(r^*) - \tau_{i,alt_j}(r^*)|$, where r^* is the resource sets in which a_j is a preferred contributors (resp. preferred recommender) agent and alt_j is the agent in $F_i \bigcup AG_i$ having the best trust (resp. honesty) value on r^* . Therefore, we compute the factor *Loss of Performance* (L_i), for an agent i , as the average of the sum of the two contributions described above:

$$L_i = \frac{L_i^{(\tau)} + L_i^{(\beta)}}{2}$$

where

$$L_i^{(\tau)} = \frac{\sum_{j \in (\bigcup_{r \in R} PC_i^r - F_i \bigcup AG_i)} (\tau_{i,j}(r^*) - \tau_{i,alt_j}(r^*))}{\|\bigcup_{r \in R} PC_i^r - F_i \bigcup AG_i\|}$$

and

$$L_i^{(\beta)} = \frac{\sum_{j \in (\bigcup_{r \in R} PR_i^r - F_i \bigcup AG_i)} (\beta_{i,j}(r^*) - \beta_{i,alt_j}(r^*))}{\|\bigcup_{r \in R} PR_i^r - F_i \bigcup AG_i\|}$$

If a_j is a preferred contributor (recommender) agent on more resources sets, r^* will be the set having the highest trust (honesty) value, then the factor $L_i^{(\tau)}$ (resp. $L_i^{(\beta)}$) is obtained by computing the average of all these contributions.

Additional Cost (C): $(F_i \cup AG_i) - (\bigcup_{r \in R} PA_i^r) \neq \emptyset$. It means that some agents belong to the set $F_i \cup AG_i$ but not to the set $\bigcup_{r \in R} PA_i^r$. In this case we measure the ratio of agents that will be never contacted by a_i to obtain help for free by computing the factor *Additional Cost* as:

$$C_i = \frac{\| F_i \cup AG_i - \bigcup_{r \in R} PA_i^r \|}{\| F_i \cup AG_i \|}$$

As a consequence, we can measure the “disadvantage” of a_i as the average of the sum of the factors L_i^τ , L_i^β and C_i as follows:

$$D_i = \frac{L_i^\tau + L_i^\beta + C_i}{3} \quad (6)$$

4.1 The FGF Algorithm

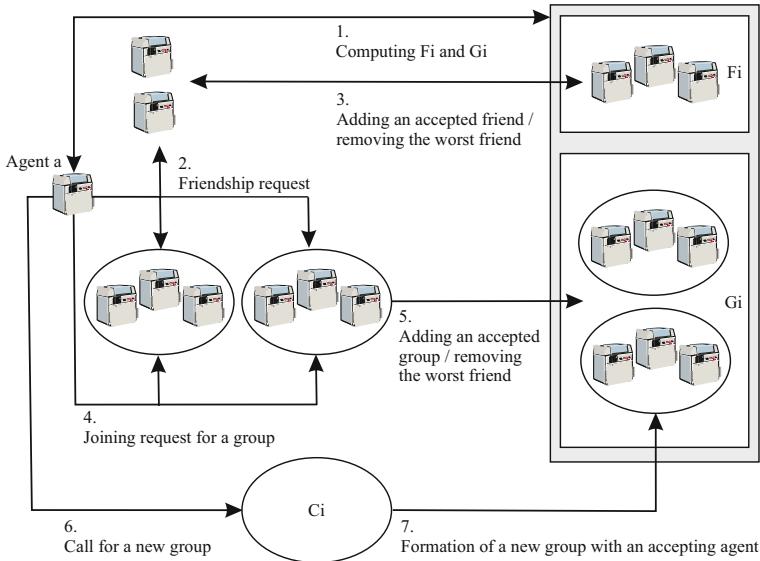
Here we provide an algorithm to be executed by each agent a_i in order to minimize, during various *epochs*, the disadvantage D_i provided by Equation 6.

We assume that the period of time between two consecutive epochs is set to a pre-fixed value T and that, in each epoch, some preferred agents join with the set $F_i \cup AG_i$ to replace agents having the worst trust or honesty values. The FGF algorithm is composed by two parts: the former (*Task A*) is a procedure designed to improve the coefficient D_i given by the Equation 6 by means of some requests which shall be initiated by the agent executing the procedure itself. The second (*Task B*) is designed to manage the requests coming from the other agents due to execution of the *Task A*.

Task A

Each agent a_i periodically executes the task A to obtain the friendship or the membership in a group of G_i for those agents belonging to the set $\bigcup_{r \in R} PA_i^r$ but not yet to the set $F_i \cup AG_i$. The Task A is composed by the following ordered sequence of steps (refer to Figure 1):

1. The sets $F_i \cup AG_i$, and $\bigcup_{r \in R} PA_i^r$ are computed (Figure 1, step 1).
2. A friendship request is sent to each agent $a_j \in (\bigcup_{r \in R} PA_i^r - F_i \cup AG_i)$ (Figure 1, step 2).
3. If a_j accepts the friendship request, then it is added to F_i (Figure 1, step 3).
4. If a_j does not accept the friendship request, then a_i executes the following steps:
 - (a) the set G_j of all the groups having a_j as a member is required by a_i to the *DF* (for a description of the *DF* see Section 2).
 - (b) for each group $g \in G_j$, a_j computes the disadvantage D_i^* .
 - (c) a joining request is sent to the group $g \in G_j$ (Figure 1, step 4) such that $D_i^* < D_i$ and D_i^* is minimum.

**Fig. 1.** The task A of an agent

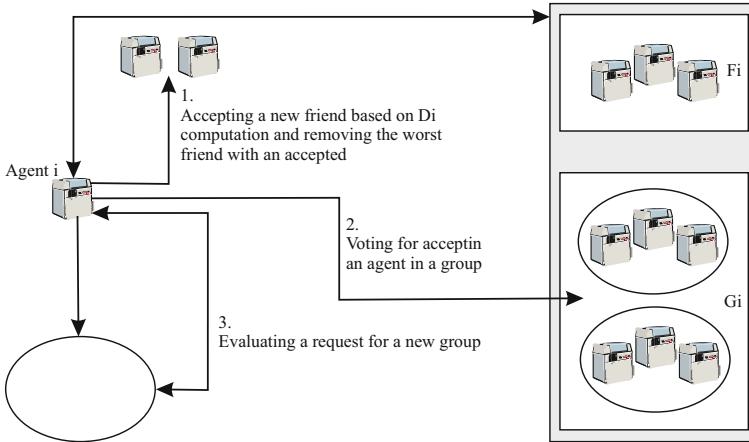
- (d) If g accepts the membership request then g is added to G_i (Figure 1, step 5), otherwise a_j is added to a set C_i .
- 5. If C_i is not empty, then a_i sends a *call for a new group* (Figure 1, step 6) to all the agents belonging to it. If some agents agree with constituting a new group, then it is formed (Figure 1, step 7) and registered to the DF.
- 6. Whenever an agent a_j is added to the set F_i , then an agent a_k , the worst friend, is removed from F_i (Figure 1, steps 3 and 5). The agent a_k is selected as follows:
 - if $a_j \in PC_i^r$, then select $a_k \notin (\bigcup_{r \in R} PA_i^r)$ having the worst trust value $\tau_{i,k}(r)$ or
 - if $a_j \in PR_i^r$, then select $a_k \notin (\bigcup_{r \in R} PA_i^r)$ having the worst honesty value $\beta_{i,k}(r)$.

Task B

Task B is a set of three subtasks designed to manage the friendship requests of the other agents, as well as the requests of joining that other agents send to groups with which a_i is joined or is a leader (administrator). The execution of one of these subtasks depends both on the role of the agent and on the nature of the received request, as specified in the following.

- *Friendship Request*. When such a request coming from an agent a_j arrives to an agent a_i then a_i :

1. computes a new disadvantage D_i^* by adding the agent a_j to the set F_i and removing an agent a_k as described by step 5 of Task A (Figure 2, Step 1).

**Fig. 2.** The task B of an agent

2. will accept the request of a_j if $D_i^* \leq D_i$. Otherwise, the request will be refused.
- *Membership Request.* When such a request coming from an agent a_j arrives to the administrator of a group g (Figure 2, Step 2), it requires a vote (positive or negative) to all the agents belonging to g . The request will be accepted by majority, otherwise it will be refused. Each agent a_k will give a positive vote if the insertion of a_j in the group g will not increase the disadvantage D_k (Figure 2, step 3).
 - *Call for a new group.* Such a request coming from an agent a_j is accepted by a_i if the insertion of a_j in the set $F_i \cup AG_i$ does not increase the disadvantage D_i (Figure 2, step 3).

5 Theoretical Results

In this Section we theoretical prove the benefits coming from the adoption of the described friendships and groups model as well as the FGF algorithm presented in Sections 3 and 4.

To this purpose, let the *Grid Capital* (GC) be the mean value of all the contributions $(1 - D_i)$ given by each agent a_i , computed as $GC = \frac{\sum_{a_i \in A} (1 - D_i)}{\|A\|}$. The *Grid Capital* increases at each iteration of the FGF algorithm because the FGF is able to optimize the global grid utility by searching new relationships among agents.

Theorem 1. *The Grid Capital increases at each iteration of the FGF algorithm.*

Proof. Taking into account the activities performed by the agent a_i at each iteration of the Task A (Section 4), we obtain that D_i : (i) increases if one or more preferred contributor or recommender agents accept its joining request; (ii)

is unvaried; (iii) decreases if one or more preferred contributor or recommender agents exit from F_i or from a group belonging to G_i . Therefore, let a_j be a preferred contributor or recommender agent in a resources set r^* . The agent a_j will exit from F_i or from a group belonging to G_i only if a_i is not one of its preferred contributors or recommenders agents and it implies that D_j decrements by 1, while D_i will increase of $\tau_{i,j}(r^*) - \tau_{i,alt_j}(r^*)$, that is lesser than 1, due to the replacement of a_j with the best alternative alt_j . Consequently, we can derive that the sum of all the agent disadvantages decreases at each iteration, while the sum of all the contributions $(1 - D_i)$ increases and this proves the Theorem 1.

In order to specify the global trustworthiness that an agent a_i receives from the whole agent community in a resources set r , we assume that: (i) let the *Merit* of an agent a_i in the resources set r (φ_i^r) be the number of agents for which a_i is preferred as contributor or recommender; (ii) let the *Expected Gain* of an agent a_i (ω_i) be the expected gain of a_i at a given step; (iii) let $P_i(x)$ be the expected value of the probability distribution such that $\omega_i = x$.

Based on this assumptions, we can state that if a_i and a_j are two agents, such that $\varphi_i^r < \varphi_j^r$ at a given iteration, then the number nc_i of clients contacting a_i for a service request falling in the resources set r will be lesser than the number nc_j of clients contacting a_j . It seems to be a reasonable consequence because if $\varphi_i^r < \varphi_j^r$, this implies that the global satisfaction of the agent community for the a_i performances is lesser than for a_j . Since the global satisfaction of the agent community is based on the clients feedbacks, it is reasonable to suppose that similarly also the clients will prefer to contact a_i instead of a_j . In other words, the choices of the clients reflect as a mirror the choices of the agents. It is true when the trustworthiness of an agent, represented by the number of other agents that consider it as a preferred interlocutor, actually capture its expertise. The mirror assumption can be considered as much valid as (i) the agent trust models are strictly based on the clients' feedbacks, similarly to that presented in Section 3 and (ii) reflects the real situation as much as the adopted trust model is able to capture the actual expertises of the agents.

Theorem 2. *For each pair of agents a_i and a_j , such that at each iteration $\varphi_i^r < \varphi_j^r$, the expected gain ω_i will be lesser than the expected gain ω_j .*

Proof. Supposing as valid the theorem statement, it implies that the number nc_i of clients contacting a_i for a service request related to r will be lesser than the number nc_j of clients contacting a_j . Consequently, the probability P_i that a contribution or a recommendation related to r can be requested by other x_i agents to a_i is lesser than the corresponding probability P_j that x_j agents can require it to a_j . Similarly, the expected number y_i of agents contacted by a_i for the resources set r is greater than the expected number y_j of agents contacted by a_j ; it is due to the high probability that the expertise of a_i is smaller than that of a_j . For sake of simplicity, suppose that a contribution or a recommendation have the same price p^* , thus at the end of the current iteration the expected gain ω_i is $u_i \cdot p + x_i \cdot p^* - y_i \cdot p^*$, that is smaller than the corresponding gain ω_j for the agent a_j and this proves the Theorem 2.

6 Experiments

In this Section we present the results of some experiments by which we study the behavior of the proposed model as a function of the parameters previously introduced. More in detail, the results shown in this Section have been obtained by a set of simulations performed on the Octave numerical tool [22] and based on a set of parameters that we summarize in Table 1.

Table 1. Simulation parameters

Simulation Parameter	Value
N_{nodes}	1000
N_{grids}	10
r sets	$\{r_1, r_2, r_3\}$
No. of HP Grids	G_1-G_4
No. of MP Grids	G_5-G_7
No. of LP Grids	G_8-G_{10}
H.P. feedbacks, (range of gen. values)	$\{0.6 \dots 0.8\}$
M.P. feedbacks, (range of gen. values)	$\{0.4 \dots 0.6\}$
L.P. feedbacks, (range of gen. values)	$\{0.1 \dots 0.4\}$
Recommendations (range of gen. values)	$\tau \pm (0.1\tau)$
Average no. of generated Feedbacks (per step)	$\left\{ \frac{N}{10} \dots \frac{N}{2} \right\}$
Average no. of generated Reccomendations (per step) ($\bar{\alpha}, \bar{\beta}, \bar{\gamma}$)	$\left\{ \frac{N}{10} \dots \frac{N}{2} \right\}$ (0.5, 0.5, 0.5)
Average no. of friends (random)	$\approx \sqrt{N}$
Average no. of groups	$\approx \frac{N}{2}$
Average level of group membership (random)	$\approx \frac{\sqrt{N}}{2}$

```

for T = 1 : END
    advance_epoch() // It serves as key for collected data
    simulate_feedbacks() // Parallel
    simulate_recc() // Parallel
    update_tau_for_all_nodes() // Parallel
    update_disadvantage_for_all_nodes() // Parallel
    for a = 1 : N
        task_A(a); // Functions of Task B are consequently invoked.
    endfor
endfor

```

Fig. 3. Simulation flow

We simulated (1) a Grid federation composed by 1000 nodes equally distributed into 10 different Grid Virtual Organizations, G_1-G_{10} , each one exposing different performance levels, i.e. High performance (HP), Middle Performance

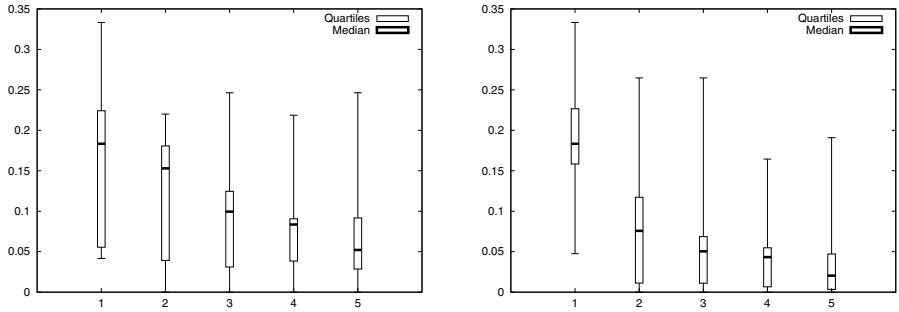


Fig. 4. Disadvantage (D). Min, 1st Quartile, Median, 3th quartile, Max. Left: $X=Y=10$ Right: $X=Y=40$, $\tau^{min} = \beta^{min} = 0.2$

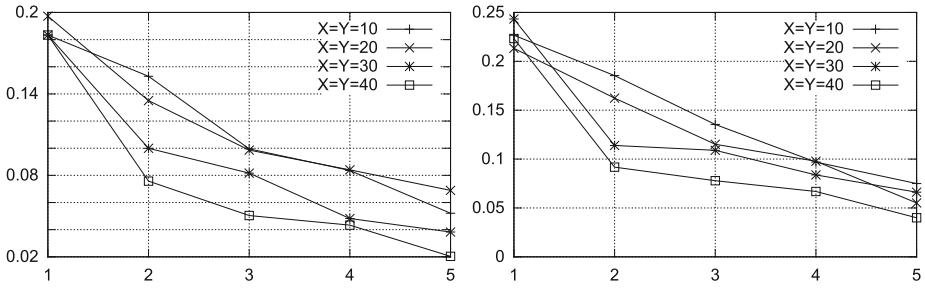


Fig. 5. Median value of disadvantage, Left: $\tau^{min} = \beta^{min} = 0.2$. Right: $\tau^{min} = \beta^{min} = 0.5$

(MP), and Low Performance (LP). The term “performance” is used in this context to indicate the level of simulated reliability of the considered Grid infrastructure or, in other words, of the related nodes providing services. Therefore, the different levels of performance are simulated by generating different values for feedbacks – and, on the consequence, different values of recommendations – for different services.

Since we were interested to confirm the theoretical results provided into the previous Section about a lowering of the Disadvantage (D) (i.e. a rices of the average $Grid\ Capital\ GC = \frac{\sum_{a_i \in A} (1 - D_i)}{\|A\|}$) caused by the execution of the FGF algorithm presented into Section 4, we did not make any distinction between resource sets among different Grids. Then we assumed, as shown into Table 1, that all the Grids have the same “bag” of resource sets, r_1, r_2, r_3 . Moreover, Table 1 also shows the average number of friends and groups. In particular, the “average level of group membership” indicates the average number of groups to which the Grid nodes have been attached during the network generation.

To let the reader better understand how we simulated the proposed model, we provide in Figure 3 a very simple listing of pseudo code, which represents

the basic simulation flow we employed for the experiments. The code is serial for that concerning the execution of *taskA*, but we have made parallel the code for which no critical races can occur, indicating them as *Parallel* into Figure 3³.

We report into Figure 4 the median values, quartiles and outliers of the Disadvantage for a set of simulations for which we set $\tau^{min} = \beta^{min} = 0.2$, i.e. the minimum value of trust and honesty to select the set *PC* and *PR* (see Section 3). We report only the first five steps of the simulation because the trend stabilizes very quickly and, although the initial groups and friends are initially random, after the first step of execution of the Task A the average disadvantage becomes very low. While on the left part of Figure 4 we present results for ($X = Y = 10$), where *X* and *Y* are the maximum size of the sets *PC* and *PR* (see Section 3), the results shown on the right concern ($X = Y = 40$). We observed that the median value of the disadvantage has a downward trend, thus confirming the theoretical results presented into the previous Section 5. More important, we remark that, as the sets *PC* and *PR* grow in size (from ($X = Y = 10$) – left part of Figure 4 to ($X = Y = 40$) – right part of Figure 4), the median (so the quartiles) assumes lower values very quickly, which is an expected behavior.

Furthermore, the behavior explained above is better described by results provided into Figure 5, on which we plotted only the median value of the Disadvantage, and *X* and *Y* ranging from 10 to 40 by steps of 10. Moreover, by comparing the results on the left part of Figure 5 with those shown in the right part of Figure 5, we observed that the more selective is the parameter τ^{min} (i.e. β^{min}) (i.e. the minimum value of trust (i.e. honesty) to put a node into the set *PC* (i.e. *PR*), the greater will be, in average, the disadvantage. Also this behavior seems to be conform with the described model and theoretical results.

Summarizing, the results of the experiments presented in this Section clearly show as the proposed model is correct and the execution of the FGF algorithm, supported by the adopted trust model, it is effectively capable to allow Grid federations to improve their provided/perceived QoS.

7 Related Work

The various aspects related to the partner/node selection and collaboration issues in the context of self-interested agents and grid systems have been dealt in a large number of models and architectures. Their overall discussion would require too much space and, therefore, the examined approaches will be only those that, to the best of our knowledge, come closest to our proposal.

In the literature different metrics have been proposed to select the most appropriate partners, for instance by exploiting local decision and models [30,37] or by promoting agent interactions to realize a distributed social control mechanism for evaluating other agents or their provided services [14]. Many of such models consider direct observations and/or communications with other agents and different criteria as trust, reputation, provided QoS, etc. In this context,

³ We used the package *parallel* provided with Gnu Octave.

the concept of *belief* can be considered as a situational awareness, and its modification can require to select the most appropriate providers for information. To this purpose, in multi-agent systems the beliefs of the agents might consider the preferred agents to obtain suitable information [2,18], where the preferred agents could be selected, for instance, on the basis of trustworthiness [2] or statistical [18] criteria. Note that these techniques usually result computational expansive [28,35] and therefore some approximate and less expensive solutions have been proposed in the context of specific beliefs [9,27].

To solve coalition formation issues among self-interested agents, negotiation mechanisms (requiring peer-to-peer communications) can be used to find the best candidates to join with. The Contract Net Protocol (CNP) [12,43] is a fully automated negotiation protocol where each agent can be an initiator or a participant of a call for proposal and where only the best participants bids on that call are selected by the initiator. This (relatively) simple partner selection mechanism might result computationally expensive on large-scale systems due to the message approach and, moreover, it does not guarantee the real execution of the “contract”. However, the CNP has been embedded into the Transportation Cooperation Net (TRACONET) [42] for a vehicle routing application according to the Foundation for Intelligent Physical Agents (FIPA) standard. Another negotiation partner selection scheme is the Adaptive Decision Making Framework (ADMF) [3]; it has been designed for systems where agents, assumed to be cooperative, share global goals to be maximized and allows a dynamic adjustment of agents relationships, although also this proposal has a low scalability in presence of large-scale systems.

In a multi-agent system the coalition formation provides to partition agents in groups in order to optimize groups and/or agents utility. The agents partition activity, might be modeled as a function game [42] involving *a*) the generation of the coalition structure, *b*) the solution of optimization problems (for each coalition) and *c*) the pay-off distribution. Activities *a* and *b* provide to search suitable partnerships from a set of cluster of agents, while the last activity distributes the coalition gain among the participants and, as important *collateral* effects, it promotes the agents’ collaboration and the stability of the coalition. Recently, some proposals adopted trust in competitive agent systems [21,39], for instance, to constitute clusters of agents [6,20] and for generating recommendations in social network contexts [13] or to detect group of actors in a competitive social community [31,32,40].

Summarizing, none of the cited proposals deal with the issue to improve the social capital of the agent community on the basis of a meritocracy criterion. On the contrary, such approaches exploit trust measures to provide an agent with suggestions about the best agents to contact as fruitful interlocutors, but without to face the issue of the possible advantages for its community. Differently, our proposal introduce a meritocratic principle in order to obtain such an advantage, also by encouraging the actors to assume correct behaviors to improve their reputation.

On the other hand, employing software agents into Grid systems has always been a subject of research [15]. Several works in the literature focused on the optimization of various aspects of Grid jobs scheduling and resources allocation. Most of them rely on the adoption of economical models as, for instance, in [7]. In [29] a strategy for optimizing the QoS into the Grid is presented. The work is based on a distributed iterative algorithm behind a mathematical model. The authors mainly deal with task optimization and resource optimization, by means of software agents. Even their goal is maximizing the global utility of the system, their approach is not distributed, indeed the different agents collaborate to perform optimization activities for the whole system. Authors of [10] analyze the interactions between Grid user agents and the Grid providers in order to maximize the whole utility of all Grid users. They propose a price-based resource allocation model by defining a *Grid User Utility* as a function of the user's allocated resources by using a nonlinear optimization theory, in order to incorporate Grid resource capacity constraint and job completion times. Nevertheless, they do not deal with the heterogeneity of resources, and do not rely on the concept of meritocracy (reliability and trust) to improve the overall QoS.

Authors of [19] combined the principles and the concepts found in social networks to design decentralized and adaptive resource discovery approach in complex Grid systems. Experimental results show as the relationship among clusters can improve the resource discovery processes, allows different resource distributions and user request patterns to a better adaptation, but the approach lacks of a component permitting to improve the social capital of the agent (node) community by improving meritocracy.

8 Conclusions and Future Work

In this paper, we presented an agent based model to optimise the global QoS of a “competitive” Grid Federation, on which computational nodes are supported by intelligent agents, which manage friendships and group memberships. Our proposal focused on the concepts of (*i*) computational *resource sets* characterising jobs in Grid Federations, (*ii*) *agent aggregation* (i.e. friendships and group memberships) as basis of collaboration between federated nodes, which, in turn, are supported by (*iii*) a *trust* model conceived to compute a unique synthetic trust measure from reliability, honesty and reputation measures. We designed an algorithm, called Friendship and Group Formation (FGF), which allows Grid nodes to select their partners (friends and group memberships) in order to improve the global QoS. For this aim the algorithm uses the trust information to compute two measures, the (*i*) *disadvantage* (D), which represents a local indication of the QoS that the single node is able to provide to the other Grid nodes, and the (*ii*) *Grid Capital* (GC), which is a global index, telling us how well the Brokers/Nodes of the Grid Federations can work together when a computational task requires an inter-Grid collaboration. The validity of the proposed model has been supported by theoretical evidences and by some experimental results, by which we have shown that the adoption of the FGF algorithm, suitably supported by the proposed trust model, the Grid Capital (which reflects the global

QoS) of the Grid Federation is effectively improved. In our ongoing research, we plan to better study the influence of several parameters characterising our model also by considering very large Grid Federations. Moreover, we plan to compare the performance of the FGF algorithm with other similar approaches which are based on aggregations and trust information. For this aim, we will possibly use ComplexSim [33,34], which is a C-based complex network parallel simulator written by some of the authors.

Acknowledgements. This work has been partially supported by the TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research and by the Program “Programma Operativo Nazionale Ricerca e Competitività” 2007-2013, Distretto Tecnologico CyberSecurity funded by the Italian Ministry of Education, University and Research.

References

1. Bandieramonte, M., Di Stefano, A., Morana, G.: An ACO inspired strategy to improve jobs scheduling in a grid environment. In: Bourgeois, A.G., Zheng, S.Q. (eds.) ICA3PP 2008. LNCS, vol. 5022, pp. 30–41. Springer, Heidelberg (2008)
2. Barber, K.S., Kim, J.: Soft security: Isolating unreliable agents from society. In: Falcone, R., Barber, S.K., Korba, L., Singh, M.P. (eds.) AAMAS 2002. LNCS (LNAI), vol. 2631, pp. 224–233. Springer, Heidelberg (2003)
3. Barber, K.S., Martin, C.E.: Dynamic reorganization of decision-making groups. In: Proc. of the 5th Int. Conf. on Autonomous Agents, pp. 513–520. ACM (2001)
4. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. ACM SIGOPS Operating Systems Review 37(5), 164–177 (2003)
5. Boghosian, B., Coveney, P., Dong, S., Finn, L., Jha, S., Karniadakis, G., Karonnis, N.: Nektar, spice and vortronics: using federated grids for large scale scientific applications. Cluster Computing 10(3), 351–364 (2007)
6. Buccafurri, F., Comi, A., Lax, G., Rosaci, D.: A trust-based approach to clustering agents on the basis of their expertise. In: Proc. of Advances in Agent and Multi-Agent Systems 2014, AMSTA 2014. ACM Press (2014)
7. Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resource management and scheduling in grid computing. Concurrency and computation: practice and experience 14(13-15), 1507–1542 (2002)
8. Buyya, R., Ranjan, R.: Special section: Federated resource management in grid and cloud computing systems. Future Generation Computer Systems 26(8), 1189–1191 (2010)
9. Chopra, S., Parikh, R., Wassermann, R.: Approximate belief revision preliminary report. Journal of IGPL (2000)
10. Chunlin, L., Layuan, L.: Multi economic agent interaction for optimizing the aggregate utility of grid users in computational grid. Applied Intelligence 25(2), 147–158 (2006)
11. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid information services for distributed resource sharing. In: 2001 Proc. 10th IEEE Int. Symp. on High Performance Distributed Computing, pp. 181–194. IEEE (2001)

12. Davis, R., Smith, R.G.: Negotiation as a metaphor for distributed problem solving. *Artificial intelligence* 20(1), 63–109 (1983)
13. De Meo, P., De Meo, A., Rosaci, D., Ursino, D.: Recommendation of reliable users, social networks and high-quality resources in a social internetworking system. *AI Communications* 24(1), 29–50 (2011)
14. Decker, K., Sycara, K., Williamson, M.: Middle-agents for the internet. In: IJCAI (1), pp. 578–583 (1997)
15. Foster, I., Jennings, N.R., Kesselman, C.: Brain meets brawn: Why grid and agents need each other. In: Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, vol. 1, pp. 8–15. IEEE Computer Society (2004)
16. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* 15(3), 200–222 (2001)
17. Franklin, S., Graesse, A.: Is it an agent, or just a program?: A taxonomy for autonomous agents. In: Jennings, N.R., Wooldridge, M.J., Müller, J.P. (eds.) ECAI-WS 1996 and ATAL 1996. LNCS, vol. 1193, Springer, Heidelberg (1997)
18. Fullam, K.K., Barber, K.S.: Using policies for information valuation to justify beliefs. In: Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 404–411. IEEE (2004)
19. Gao, L., Ding, Y.S., Ying, H.: An adaptive social network-inspired approach to resource discovery for the complex grid systems. *International Journal of General Systems* 35(3), 347–360 (2006)
20. Garruzzo, S., Rosaci, D.: Agent clustering based on semantic negotiation. *ACM Transactions on Autonomous and Adaptive Systems* 3(2) (2008)
21. Garruzzo, S., Rosaci, D., Sarné, G.M.L.: Integrating trust measures in multi-agent systems. *International Journal of Intelligent Systems* 27(1), 1–15 (2012)
22. GNU Octave, <http://www.gnu.org/software/octave/>
23. Yuan, L.-L., Zeng, G.-S., Jiang, L.-L., Jiang, C.-J.: Dynamic level scheduling based on trust model in grid computing. *Chinese Journal of Computers* 7, 23 (2006)
24. Huynh, T.D.: Trust and reputation in open multi-agent systems. PhD thesis, University of Southampton (2006)
25. Katia, L., Eduardo, H., Ignacio, M.L.: A decentralized model for scheduling independent tasks in federated grids. *Future Generation Computer Systems* 25(8), 840–852 (2009)
26. Kivity, A., Kamay, Y., Laor, D., Lublin, U., Liguori, A.: KVM: the linux virtual machine monitor. In: Proc. of the Linux Symp., vol. 1, pp. 225–230 (2007)
27. Koller, D., Lerner, U., Angelov, D.: A general algorithm for approximate inference and its application to hybrid bayes nets. In: Proc. of the 15th conf. on Uncertainty in artificial intell., pp. 324–333. Morgan Kaufmann Pub. (1999)
28. Lerner, U., Segal, E., Koller, D.: Exact inference in networks with discrete children of continuous parents. In: Proc. of the 17th Conf. on Uncertainty in Artificial Intell., pp. 319–328. Morgan Kaufmann Pub. (2001)
29. Li, C., Li, L.: Utility-based qos optimisation strategy for multi-criteria scheduling on the grid. *Journal of Parallel and Distributed Computing* 67(2), 142–153 (2007)
30. Maximilien, E.M., Singh, M.P.: Agent-based trust model involving multiple qualities. In: Proc. of the 4th Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 519–526. ACM (2005)
31. Messina, F., Pappalardo, G., Rosaci, D., Santoro, C., Sarné, G.M.L.: A distributed agent-based approach for supporting group formation in P2P e-learning. In: Baldoni, M., Baroglio, C., Boella, G., Micalizio, R. (eds.) AI*IA 2013. LNCS, vol. 8249, pp. 312–323. Springer, Heidelberg (2013)

32. Messina, F., Pappalardo, G., Rosaci, D., Santoro, C., Sarné, G.M.L.: HySoN: A distributed agent-based protocol for group formation in online social networks. In: Klusch, M., Thimm, M., Paprzycki, M. (eds.) MATES 2013. LNCS, vol. 8076, pp. 320–333. Springer, Heidelberg (2013)
33. Messina, F., Pappalardo, G., Santoro, C.: Complexsim: An smp-aware complex network simulation framework. In: 2012 Sixth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pp. 861–866. IEEE (2012), doi:10.1109/CISIS.2012.102
34. Messina, F., Pappalardo, G., Santoro, C.: Complexsim: a flexible simulation platform for complex systems. International Journal of Simulation and Process Modelling 8(4), 202–211 (2013), doi:10.1504/IJSPM.2013.059417
35. Nebel, B.: How hard is it to revise a belief base? Springer (1998)
36. Papazoglou, M.P.: Service-oriented computing: Concepts, characteristics and directions. In: Proc. of the 4th Int. Conf. on Web Information Systems Engineering, WISE 2003, pp. 3–12. IEEE (2003)
37. Park, J., Barber, K.S.: Information quality assurance by lazy exploration of information source combinations space in open multi-agent systems. J. UCS 11(1), 193–209 (2005)
38. Ranjan, R., Chan, L., Harwood, A., Karunasekera, S., Buyya, R.: Decentralised resource discovery service for large scale federated grids. In: IEEE International Conference on e-Science and Grid Computing, pp. 379–387. IEEE (2007)
39. Rosaci, D.: Trust measures for competitive agents. Knowledge-based Systems (KBS) 28(46), 38–46 (2012)
40. Rosaci, D., Sarné, G.M.L.: Matching users with groups in social networks. In: Zavoral, F., Jung, J.J., Badica, C. (eds.) IDC 2013. SCI, vol. 511, pp. 45–54. Springer, Heidelberg (2013)
41. Rosenblum, M.: Vmwar's virtual platform. In: Proceedings of hot chips, pp. 185–196 (1999)
42. Sandholm, T.W., Lesser, V.R.T.: Coalitions among computationally bounded agents. Artificial intelligence 94(1), 99–137 (1997)
43. Smith, R.G., Davis, R.: Frameworks for cooperation in distributed problem solving. IEEE Trans. on Systems, Man and Cybernetics 11(1), 61–70 (1981)
44. Vázquez, T., Huedo, E., Montero, R.S., Llorente, I.M.: Evaluation of a utility computing model based on the federation of grid infrastructures. In: Kermarrec, A.-M., Bougé, L., Priol, T. (eds.) Euro-Par 2007. LNCS, vol. 4641, pp. 372–381. Springer, Heidelberg (2007)
45. Yu, J., Buyya, R.: A novel architecture for realizing grid workflow using tuple spaces. In: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, pp. 119–128. IEEE (2004)
46. Yu, J., Buyya, R.: A taxonomy of workflow management systems for grid computing. Journal of Grid Computing 3(3-4), 171–200 (2005)

Towards a Formal Specification of SLAs with Compensations*

Carlos Müller, Antonio M. Gutiérrez, Octavio Martín-Díaz, Manuel Resinas,
Pablo Fernández, and Antonio Ruiz-Cortés

University of Sevilla, Sevilla, Spain

Abstract. In Cooperative Information Systems, service level agreements (SLA) can be used to describe the rights and obligations of parties involved in the transactions (typically the service consumer and the service provider); amongst other information, SLA could define guarantees associated with the idea of service level objectives (SLOs) that normally represent key performance indicators of either the consumer or the provider. In case the guaranteee is under-fulfilled or over-fulfilled SLAs could also define some compensations (i.e. penalties or rewards). In such a context, during the last years there have been important steps towards the automation of the management of SLAs, however the formalization of compensations in SLAs still remains as an important challenge.

In this paper we aim to provide a characterization model to create SLAs with compensations; specifically, the main contributions are twofold: (i) the conceptualization of the Compensation Function to express consistently penalties and rewards and (ii) a model for Compensable Guarantees that associate SLOs with Compensation Functions. This formalization models aim to establish a foundation to elaborate tools that could provide an automated support to the modeling and analysis of SLAs with compensations. Additionally, in order to validate our approach, we model and analyze a set of guarantee terms from three real world examples of SLAs and our formalization proves to be useful for detecting mistakes that are typically derived from the manual specification of SLAs in natural language.

1 Introduction

The shift from product to services in the industry is a major trend for developed countries. In such a context this evolution implies the creation of a network of dependable organizations that exchange services and create cooperative information systems (CIS) to gain business value. For instance, in a cloud scenario a Software as a Service (SaaS) provider may use several Infrastructure as a Service (IaaS) providers such as Amazon Elastic cloud (EC2)¹ and the Google Cloud

* This work was partially supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants P12-TIC-1867, TIN2012-32273, TIC-5906 and IPT-2013-0890-3)

¹ Available at <http://aws.amazon.com/ec2/>

Storage service², to offer a service combination of a virtual machine with support for data persistence.

In this cooperative environment there is a craving for guarantees that support a reliable service consumption and cooperation, and service level agreements (SLAs) represent a first-class citizen to describe the parties rights and obligations. Specifically, SLAs are composed by different terms that typically define guarantees associated with a certain service level objective (SLOs) and they should be enforced by one party (the guarantor) to another party (the beneficiary); in most cases the former correspond to the service provider, and the latter to the service consumer. Additionally, real world SLAs usually include a set of compensations that represent the consequences of underfulfilling (penalties) or overfulfilling (rewards) the SLOs. We coin the concept of Compensable SLAs referring to such SLAs that include at least a compensation action, either a penalty or a reward. An example of the importance of this kind of SLAs is the cost of cloud service unavailability amounting to more than 70 million USDs based on hourly costs, by providers such as Amazon and Microsoft from 2007 to 2012 [8].

In this scenario it is important to note that in spite intraorganization modeling has been extensively studied with concepts of KPI[9] or PPI[5], there is a lack of a formal model to specify SLAs with compensation mechanisms. In fact, there is a strong relationship between the KPI of an organization and its commitments in terms of SLOs identified within an SLA.

Specifically, this work is focused on the modeling of SLAs with compensations with two main research goals: (i) a formal definition of different kinds of compensations, either penalties or rewards, and (ii) the checking of some desirable properties to automate the analysis of compensations. This analysis would represent important benefits for both consumers and providers in CIS: On the one hand, service providers could automate the optimization of the provision of services based on the compensations involved; on the other hand, service consumer could automate the analysis of guarantees in the SLA to understand its risk.

Our approach is grounded on the novel definition of Compensation Function (CF) that is inspired in the concept of penalty function introduced by Leitner et al. [11] and it has been extended to include the notion of rewards and to be aligned with the current most prominent SLA specification (WS-Agreement [1]). In addition, we also extend the notion of consistence and validity proposed in [14] in order to include compensations by means of a formal definition of properties that could be automatically checked. Our proposal has been successfully applied to model three real world SLAs that define different types of compensations and to detect some potential inconsistencies.

This paper is structured as follows: Section 2 introduces the motivating scenarios of three real world SLAs; in Section 3 we present the conceptualization of the Compensation Function. In Section 4 we present the relationships between the SLO and Compensation Function to formalize the concept of Compensable Guarantees and model the different examples presented in the motivating scenarios.

² Available at <http://cloud.google.com/products/cloud-storage>

In Section 5 we analyze the literature to identify related approaches to deal with compensations. Finally, in Section 6 we outline some conclusions and future work.

2 Motivating Scenarios

SLAs are widely used in the industry in situations where consumers and providers need or desire to explicitly express certain guarantees over the service transactions. These guarantees are typically tied to certain consequences in terms of penalties and rewards depending whether the guarantee is underfulfilled or overfulfilled; we commonly refer to these consequences as compensations.

In this section we motivate the need for formal compensable agreements with three real world scenarios that include both computing services and human-driven services. In all cases, there is a strong need to express compensations related to the guarantees defined. In the rest of the section we introduce the scenario and present an example of compensation identified in its SLA.

2.1 AWS EC2 SLA

Amazon Web Services (AWS) is a service catalogue that has boosted the idea of cloud computing in the industry; amongst them, the Elastic Computing Cloud (EC2) represents a widely used Infrastructure as a Service. The aim of EC2 is to provide a scalable infrastructure to organizations that have variable needs or they need to grow seamlessly without the investment for an internal data center. In this context, the reliability of a virtualized infrastructure represents a key point for IaaS consumers in order to choose a service like AWS EC2.

As a consequence, Amazon has explicitly published an SLA for EC2³ that is based on the idea of Monthly Uptime Percentage (MUP); this element, characterizes a guarantee over the availability of the virtual resources requested. Specifically, the consequences of failing a certain MUP is defined by Amazon in two levels: in case the MUP drops below 99.95 percent and in case the MUP drops below 99 percent. Figure 1 depicts the actual penalty function [10] of this scenario that is defined as a percent of discount in the next billing cycle a.k.a Service Credit Percentage (SCP). Note that in such Figure a dark point denotes the inclusion of the service property value in the interval, gray points means the value exclusion.

2.2 Telecomm SLA

The regional Government of Andalusia in Spain outsources the installation and management of telecommunication networks. Specifically, the demanded services include issues such as: project managing, interventions, network maintenance, installations, and logistics. An SLA⁴ is specified by the regional government including some penalties for the services provider.

³ Available at <http://aws.amazon.com/es/ec2/sla/>

⁴ Available at <http://goo.gl/WIke8y>

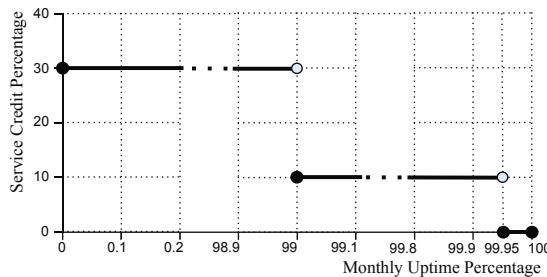


Fig. 1. Example AWS EC2: Penalties for Amazon as provider

Two examples of terms with penalties have been selected from the agreement. On the one hand, the term shown in the example ARG-1 of Figure 2 demands that 90% of interventions must be solved (cf. solid vertical line in the graph). However, as depicted in the table some penalties apply for a range of values that fulfill such a demand. Specifically, the table establishes that if the service provider solve more than 95% of interventions no penalties apply, but some bill penalties apply from 90% to 95% of interventions solved. This situation could imply a definition error that must be tackled in our proposal. On the other hand, the term shown in the example ARG-2 (cf. Figure 3) has not the problem of ARG-1 because it demands solving 95% of urgent interventions and any underfulfillment involve penalties.

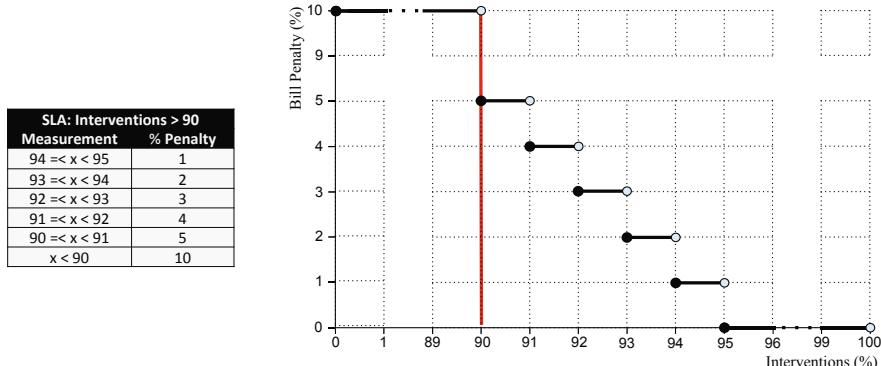


Fig. 2. Example ARG-1: Penalties for the Andalusian Regional Government

2.3 GNWT SLA

The Government of the Northwest Territories (GNWT) of Canada outsources the IT support. Specifically, the demanded services include issues related to: reporting, user support, problem correction, application enhancement, process

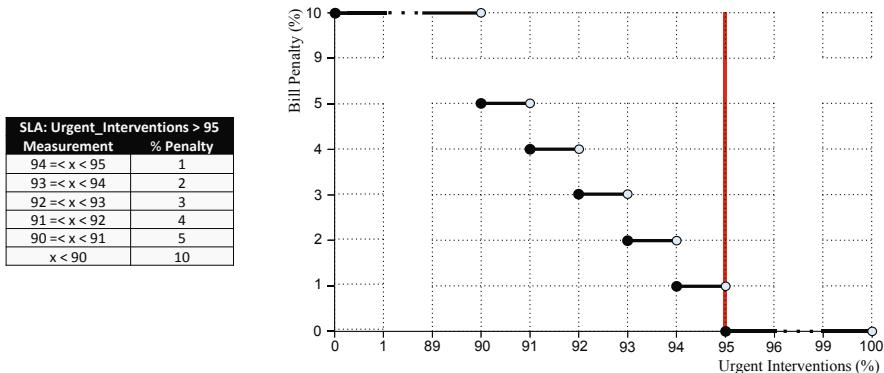


Fig. 3. Example ARG-2: Penalties for the Andalusian Regional Government

and application improvement, and other services. They provide a template for establishing an SLA with an external vendor providing the mentioned kind of IT support with the desired service levels and penalties and rewards for the parties.

Four examples of terms with at least a penalty or a reward have been extracted from its SLA template⁵. In the example GNWT-1 included in Figure 4 the GNWT demands the delivery of quarterly reports. The government must receive such reports not less than five days before scheduled review meetings under a penalty of 5% of monthly invoice for the IT support provider.

Example GNWT-2 of Figure 4 depicts specific times for different milestones that take place in the resolution of problems that have made a critical application function unusable or unavailable and no workaround exists (severity 1 code). Specifically, an initial response should be received within 15 minutes, an estimation response should be ready in 2 hours, subsequent responses are expected every 30 minutes, and the problem must be resolved within 4 hours. In this case, a reward for the provider applies if all problems are resolved in less than 2 hours, and a penalty for the provider applies if any of them is resolved in more than 4 hours. An additional clause rewarding than no problem is older than 60 days is included as example GNWT-3 of Figure 4 shows. Note that in previous examples of Figures 1, 2, and 3 just a penalty for a party is established and thus we consider them as *half-compensated* terms. However, in this term and other such as the example GNWT-4 of Figure 4 not only penalties for the provider are established (cf. Figure 5a), but also rewards (cf. Figure 5b). In our proposal we tackle a *full-compensated* joint modeling for penalties and rewards.

This SLA also includes a term relating the scheduled project delivery and the real project delivery that is shown in example GNWT-4 of Figure 4. This term includes a reward for the provider if the elapsed days until delivery is less than 20% lesser than planned but also a penalty for the provider if the elapsed days until delivery is exactly 20% greater than planned (cf. 120% value in Figure 6).

⁵ Available at <http://www.fin.gov.nt.ca/ocio/sim/sdlc/3/resources/sla.htm>

Type	Measurement	Penalty
Quarterly Status Report	Delivered at quarterly intervals and not less than five business days before scheduled review meeting	5% of monthly invoice

Example GNWT-1

Severity Code	Initial Response	Estimation Response	Subsequent Responses	Resolution
1	15 minutes	2 hours	Every 30 min.	4 hours
Type		Measurement		
Severity 1 Resolution			All Severity 1 problems are resolved in less than 2 hours.	10% of monthly fees
One or more Severity 1 problems are resolved in over 4 hours.			NA	10% of monthly fees

Example GNWT-2

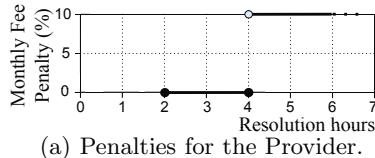
Type	Measurement	Reward	Penalty
Maximum Problem Aging	No problem is older than 60 days.	5% of monthly fees	NA

Example GNWT-3

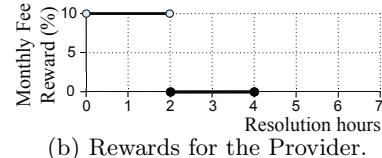
Type	Measurement	Reward*	Penalty
Project Delivery	Total elapsed days until delivery is more than 20% greater than planned.	NA	10% of the amount invoiced for the project.
	Total elapsed days until delivery is 20% less than planned.	5% of the amount invoiced for the project.	NA

Example GNWT-4

Fig. 4. Compensations actions extracted from the SLA of GNWT



(a) Penalties for the Provider.



(b) Rewards for the Provider.

Fig. 5. Penalties and rewards extracted from the example GNWT-2

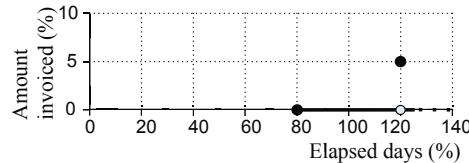


Fig. 6. Penalties extracted from the example GNWT-4

In this case the penalty could be wrong, because it would apply for any value more than 20% greater than planned and not only such exact value. Such a problem would be tackled in our proposal.

3 Compensation Functions

Compensation functions are defined over services properties in the context of a guarantee satisfied by a guarantor to a beneficiary. Specifically, they associate two types of compensations depending on the subject and recipient of the compensations: on the one hand, a penalty represents a compensation from the guarantor to the beneficiary and, on the other hand a reward represents a compensation from the beneficiary to the guarantor.

In this section we formalize the concept of Compensation Function in order to analyse some interesting properties that would support a consistency check of compensable guarantees. Following we present the formalization by means of a set of supporting core definitions:

Definition 1 (Service Property Values). *The set SP_{sp} denotes the set of all possible values of a service property sp ($SP_{sp} = \{v_1, \dots, v_n\}$).*

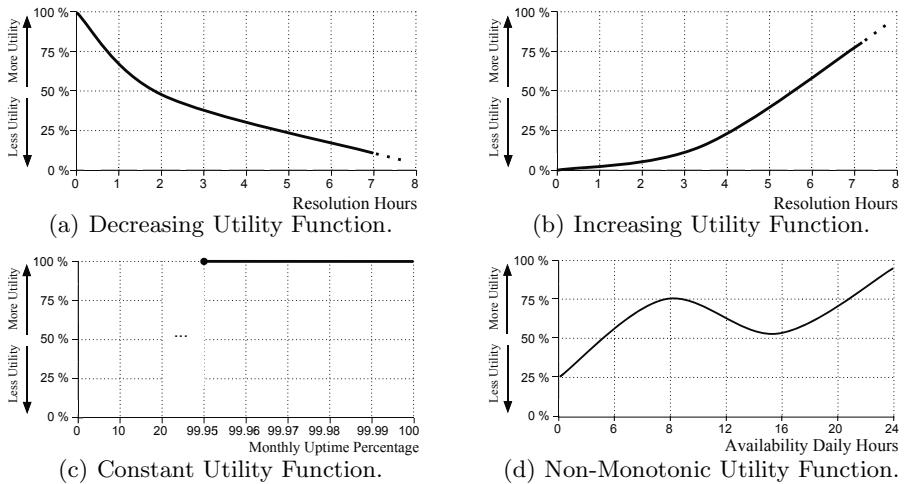
In the examples of Section 2 we find several service properties such as, MUP, interventions, and urgent interventions, with the following finite set of service property values $SP_{sp} = \{0, \dots, 100\}$; and others with an infinite set of service property values such as, resolution hours, or elapsed days $SP_{sp} = \{0, \dots, \infty\}$.

Definition 2 (Utility Function). *An Utility Function for a certain service property sp , denoted by UF_{sp} , is a function from SP to \mathbb{R} that associates a utility to each of the values; i.e. it defines which service properties values SP_{sp} are more interesting for a given party.*

Utility functions are typically complementary in case of guarantor and beneficiary. Since they are normally private, we should guess the utility function behind the examples of Section 2. For instance, Figure 7 includes two utility functions for the resolution hours service property of GNWT-2 example (described in Figure 4), an utility function for Availability warranty from Amazon EC2 and an utility function for availability daily hours which aims optimizing different customer goals, such as a common office time or fully 24 hours per day availability (from horizontal demand in [3]). As shown in the Figure, the parties may define different kinds of utility functions, namely, decreasing, increasing, constant, or non-monotonic.

Definition 3 (Utility Precedence). *Let v_1 and v_2 be values of the service property values SP_{sp} of a service property sp , and UF_{sp} a utility function defined on the same service property; a precedence relation called utility precedence is defined on SP_{sp} by UF_{sp} . Thus, we denote that v_1 is less interesting than v_2 by $v_1 \prec v_2$.*

For instance, the example of Figure 7(a) may represent the utility of the beneficiary of the GNWT-2 (described in Figure 4). In the utility precedence defined for such a utility function, the higher value of resolution hours, the less interesting for the beneficiary (e.g. $4 \prec 2$). On the other hand, the example of Figure 7(b) may represent the utility of the guarantor in the same scenario.

**Fig. 7.** kinds of utility functions

In this case, the utility precedence defined for such a utility function establishes that a lesser value of resolution hours is less interesting for the guarantor (e.g. $2 \prec 4$).

Definition 4 (Compensation Function). *A compensation function for a given service property sp , denoted by CF_{sp} , is a function from SP to \mathbb{R} that associates a compensation to each of the values. Similarly to utility functions, the compensation functions can be either decreasing, or increasing, or constant, or non-monotonic.*

As a normalized convention that is aligned with related works [10,11] we establish a positive compensation as penalties (that should be compensated from the guarantor to the beneficiary) and negative compensations as rewards (i.e. beneficiary should compensate guarantor).

Figure 8 shows an example of increasing compensation function taken from the example GNWT-2. The function denotes: (1) a penalty for the guarantor if the problems are solved in more than 4 hours; (2) a reward for the guarantor if problems are solved in less than 2 hours; and (3) no compensation applies in problems are solved from 2 to 4 hours, inclusive.

It is important to note that compensation functions could only include penalties or rewards. For instance, Examples ARG-1 and ARG-2 of Figures 2 and 3 just include penalties but not rewards.

Definition 5 (Compensation Regions). *A compensation function for a given service property sp CF_{sp} defines up to three compensation regions, namely: penalized, rewarded, and neutral.*

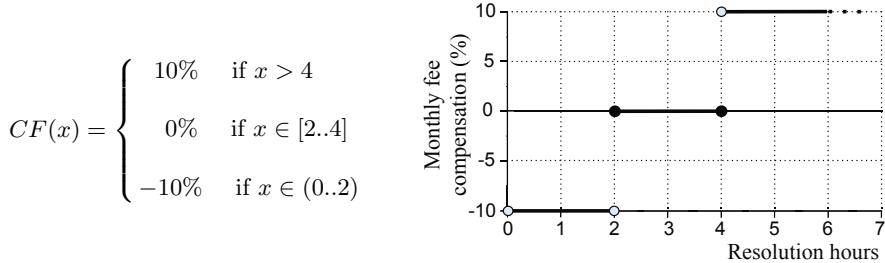


Fig. 8. Compensation function corresponding to example GNWT-2

$$\begin{aligned} Penalized(CF_{sp}) &= \{v_i \in SP_{sp} : CF_{sp}(v_i) > 0\} \\ Neutral(CF_{sp}) &= \{v_i \in SP_{sp} : CF_{sp}(v_i) = 0\} \\ Rewarded(CF_{sp}) &= \{v_i \in SP_{sp} : CF_{sp}(v_i) < 0\} \end{aligned}$$

Figure 9 shows these three potential subsets. Thus, $\forall v_i < a$ in Figure 9 v_i is a rewarded value. $\forall v_i > b$ in Figure 9, v_i is a penalized value. And $\forall v_i, a \leq v_i \leq b$ in Figure 9, v_i is a neutral value. Since omission means a lack of compensation in natural language, when there is not explicit definition of a penalty or a reward, we consider a unique subset of neutral values.

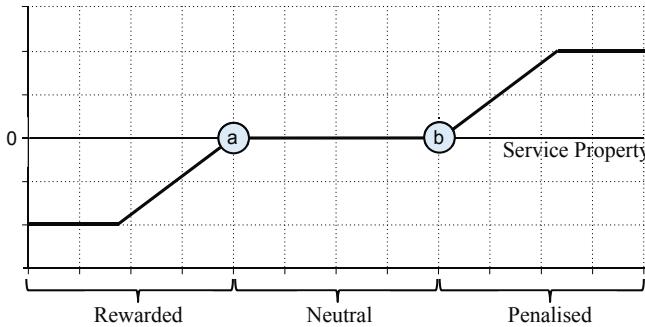


Fig. 9. A generic example of increasing compensation function

3.1 Validity of Compensation Functions

We formalize the validity of compensation functions as a property related with the consistence in terms of utility and the saturability of compensations. Consequently, for the sake of the clarity we divide this validity in two separate subproperties (Consistent and Saturated) that are defined next.

Property 1 (Consistent_{CF}). A compensation function CF_{sp} is said to be consistent if the compensation for a less interesting value of service property is less

or equal than the compensation for a more interesting value according with the utility precedence defined by the Utility function of the beneficiary. Thus, compensation and utility functions could be covariant if both are increasing or decreasing functions; and contravariant otherwise.

$$\text{Consistent}_{CF}(CF_{sp}) \iff \forall v_1, v_2 \in SP \cdot v_1 \preceq v_2 \Rightarrow CF_{sp}(v_1) \geq CF_{sp}(v_2)$$

Figures 1 and 8 depict a decreasing and an increasing consistent compensation function, respectively. However, example GNWT-4 depicted in Figure 10 is not consistent with the probable utility precedence derived from a monotonically increasing utility function of the beneficiary.

Property 2 (Saturated). Let CF_{sp} a compensation function, it is said to be saturated if there exist two values (v_{min} and v_{max}) for the service property, that delimit the higher compensation, either penalty or reward.

$$\begin{aligned} \text{Saturated}(CF_{sp}) \iff & \forall v_i \in SP, \exists v_{max}, v_{min} \in SP \\ & CF_{sp}(v_i) \leq CF_{sp}(v_{max}) \wedge CF_{sp}(v_i) \geq CF_{sp}(v_{min}) \end{aligned}$$

This property prevents the definition of infinite compensations that should be avoided in real scenarios; consequently (and as it was expected) all the different examples of compensation functions described in the paper are saturated since they correspond with real world examples.

At this point, based on the previous properties we can develop a further formalization of the validity:

Property 3 (Valid_{CF}). Let CF_{sp} a compensation function, it is said to be valid if it is both consistent and saturated.

$$\text{Valid}_{CF}(CF_{sp}) \iff \text{Consistent}_{CF}(CF_{sp}) \wedge \text{Saturated}(CF_{sp})$$

The compensation functions of Figures 1 and 8 are valid. On the contrary, Figure 10 shows the compensation function of GNWT-4 example that is not consistent and therefore, not valid.

4 Compensable SLA

SLAs specify the rights and responsibilities of the different parties involved in a service consumption. Amongst others, a key element part of SLAs are the guarantee terms [1] that are typically defined over a service level objective. Based on this conceptualization, we coin the concept of Compensable Guarantees to those which include a compensation function and subsequently, Compensable SLAs represent a type of SLA that includes at least one Compensable Guarantees.

In this context, while a guarantee term is defined around the idea of a SLO that should be guaranteed by a guarantor to a beneficiary (e.g. *Response Time* $< 100ms$ or *Monthly Uptime Percentage* $\geq 99.95\%$); a compensable guarantee term

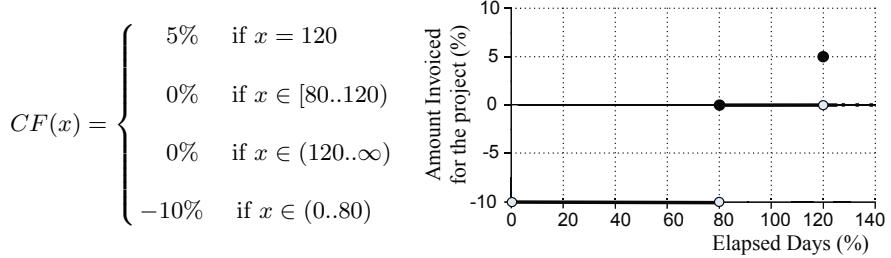


Fig. 10. Not valid Compensation function of example GNWT-4

includes the concept of penalties and rewards to compensate the underfulfillment or overfulfillment of the SLO, respectively.

Based on the formalization of the Compensation Function presented in Section 3, next we provide the formal definitions of Compensable Guarantees and SLAs. In addition, we introduce a set of properties to analyse Compensable Guarantees to extend the notion of consistence and validity in SLAs (presented in [14]) in order to ground the development of supporting tooling to help in the modeling of valid Compensable SLAs.

Definition 6 (Service Level Objective). An SLO_{sp} is a valid⁶ assertion defined over a service property sp .

Examples of SLOs include $Response\ Time < 100ms$ or $Monthly\ Uptime\ Percentage \geq 99.95\%$.

Definition 7 (Fulfillment Regions). The assertion defined by an SLO_{sp} determines two regions over the values of the service properties they dealt with, namely fulfilled and unfulfilled. These regions are delimited by the threshold Th_{SLO} .

$$\begin{aligned} Fulfilled(SLO_{sp}) &= \{v_i \in SP_{sp} \cdot SLO_{sp}\} \\ Unfulfilled(SLO_{sp}) &= \{v_i \in SP_{sp} \cdot \neg SLO_{sp}\} \end{aligned}$$

Definition 8 (Compensable Guarantee). A compensable guarantee CG_{sp} is a two-tuple of the form (SLO_{sp}, CF_{sp}) in which SLO_{sp} is a service level objective and CF_{sp} is a compensation function that are defined over the same service property sp .

$$CG_{sp} = \langle CF_{sp}, SLO_{sp} \rangle$$

Figure 11 shows a typical compensation function (defined over an increasing utility precedence) that depicts the relationships between the fulfillment regions delimited by the SLO and the compensation regions defined by the CF (c.f. Section 3). Moreover, as shown in this figure, it is important to highlight that

⁶ A formal validity criteria for SLOs is presented in [14].

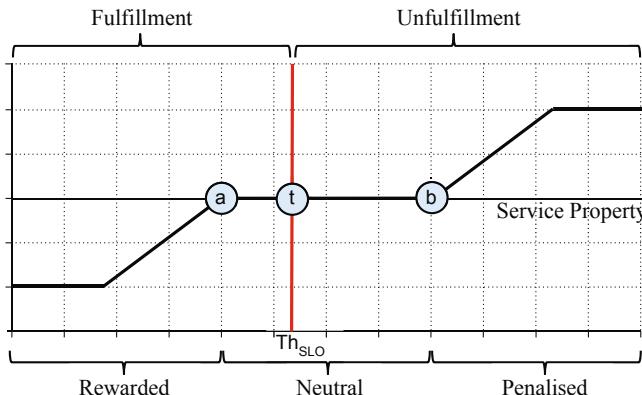


Fig. 11. A generic example of compensable guarantee showing the fulfillment and compensable regions

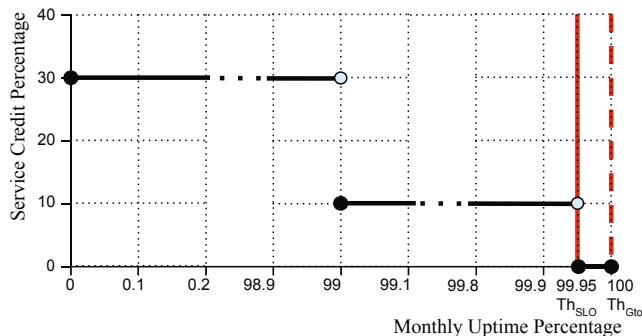
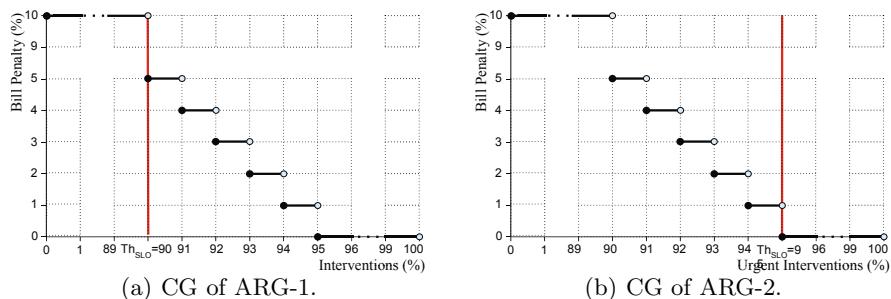
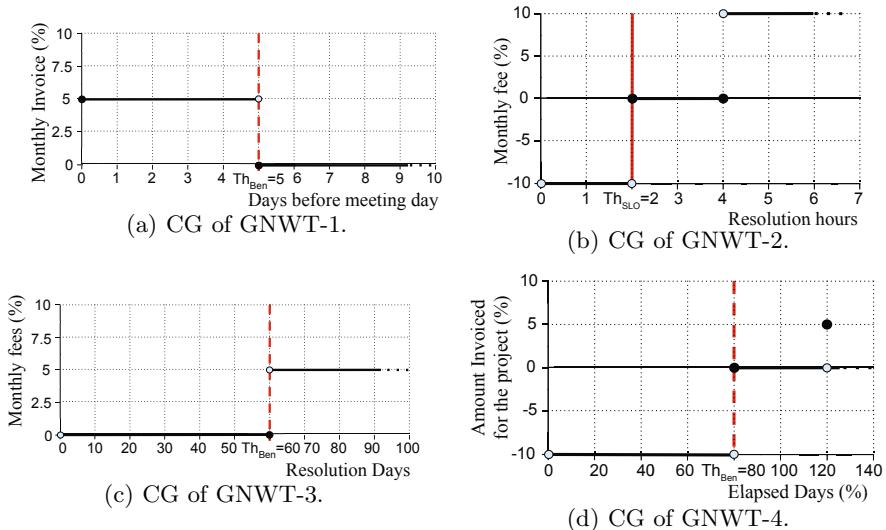
fulfillment regions are not necessarily coupled with compensation regions; specifically, figure exemplifies a case having neutral service properties values between a and b without any compensation: service properties values between t and b are unfulfilled but not penalized, and similarly service properties values between a and t are fulfilled but not rewarded. In addition, figure shows how threshold Th_{SLO} delimits the fulfillment from the unfulfillment values.

Following our formalization of compensable guarantees, Figures 12, 13, and 14 present our modeling of the compensable guarantee terms identified in the three real world SLAs found in the scenarios presented in Section 2. Each example, corresponds with a guarantee term showing the specific compensation function (as a black line) along with the Th_{SLO} (as a solid vertical line) derived from the SLO; in case there is no SLO explicit, we have inferred a threshold (Th_{Gtor} or Th_{Ben}) depending on whether the SLA was specified by the guarantor (AWS EC2 scenario) or the beneficiary (GNWT and Telecomm. SLA scenarios); these inferred thresholds are depicted as discontinuous lines.

4.1 Validity of Compensable Guarantees

The validity a compensable guarantee is derived from the coherence between its SLO and its CF. Consequently, in this section we formalize the validity criteria that is established upon the concept of validity for compensation functions and the consistency between its SLO and CF. This formalization extends the notion of validity and consistence presented in [14].

Property 4 (Consistent). A compensable Guarantee CG_{sp} is said to be consistent if there is at least one fulfilled and neutral value (that would be Th_{SLO}) and the fulfillment regions are coherent with compensation regions: the fulfilled values are either neutral or rewarded and, complementary, the unfulfilled values are either neutral or penalized.

**Fig. 12.** Example from AWS EC2 SLA**Fig. 13.** Compensable Guarantees of the ARG examples**Fig. 14.** Compensable Guarantees of GNWT examples

$$\begin{aligned}
Consistent_{CG}(CG_{sp}) \iff & Fulfilled(CG_{sp}.SLO) \cap Neutral(CG_{sp}.CF) \neq \emptyset \wedge \\
& Fulfilled(CG_{sp}.SLO) \subset Neutral(CG_{sp}.CF) \cup Rewarded(CG_{sp}.CF) \wedge \\
& Unfulfilled(CG_{sp}.SLO) \subset Neutral(CG_{sp}.CF) \cup Penalized(CG_{sp}.CF)
\end{aligned}$$

In this point, based on the previous properties we can develop a further formalization of the validity for a compensable guarantee:

Property 5 (Valid). Let CG_{sp} a compensable guarantee, it is said to be valid if it is consistent and it contains a valid CF.

$$Valid_{CG}(CG_{sp}) \iff Valid_{CF}(CG_{sp}.CF) \wedge Consistent_{CG}(CG_{sp})$$

Table 1. Comparative analysis of examples

	Consistent _{CF}	Saturated _{CF}	Valid _{CF}	Consistent _{CG}	Valid _{CG}
AWS EC2	✓	✓	✓	✓	✓
ARG-1	✓	✓	✓	✗	✗
ARG-2	✓	✓	✓	✓	✓
GNWT-1	✓	✓	✓	✓	✓
GNWT-2	✓	✓	✓	✓	✓
GNWT-3	✓	✓	✓	✓	✓
GNWT-4	✗	✓	✗	✓	✗

Table 1 presents a comparative study of the different examples (Figures 14-12) over the different properties defined. Based on this analysis we highlight the following insights:

- Example ARG-1 of Figure 13 includes a SLO that represents a *Th* with a penalty; consequently, this example does not satisfy the $Consistent_{CG}$ property with an invalid threshold and therefore the guarantee is not $Valid_{CG}$.
- GNWT-4 of Figure 14 represents a case (elapsed days) where we can induce a monotonic increasing Utility Function for the beneficiary; based on this utility precedence, the compensation function is not consistent so we can infer that this case represents a human mistake in the SLA derived from the usage of natural language.

5 Related Work

In this section we extend our initial analysis with a further exploration over relevant research papers that include the idea of SLAs with penalties and/or rewards. Table 2 shows a comparative study of the different properties identified over the examples of the papers.

The proposal of Leitner et al. in [10] formalizes the problem of finding the optimal set of adaptations, which minimizes the total costs arising from SLA violations and the adaptations to prevent them. In this work, a model for penalty functions is presented; this formalizations has been the starting point of our motivational scenario description presented in Section 2 and consequently, our approach represents an extension to this model in order to develop a complete formalization for Compensable Guarantees and SLAs. Based on our model, we have studied 4 examples (page 2) included presented in [10] relating to the cost of violations of one service property, namely: time to offer, order fulfillment time, process lead time, and cost compliance. In [11] the same authors present an approach for optimally scheduling incoming requests to virtual computing resources in the cloud, so that the sum of payments for resources and loss incurred by SLA violations is minimized. The studied example (page 3) includes a linear penalty function with two point of discontinuities. The example relates the penalty with a service property representing the duration of requests to virtual computing resources in the cloud.

Table 2. Properties analysis in Examples found in the papers

		Consistent _{CF}	Saturated _{CF}	Valid _{CF}	Consistent _{CG}	Valid _{CG}
Leitner et al. [10]	TimeToOffer<=2 (Pag.2)	✓	✓	✓	✓	✓
	OrderFulfillment<=5 (Pag.2)	✓	✓	✓	✓	✓
	ProcessLeadTime<=6 (Pag.2)	✓	✓	✓	✓	✓
	CostCompliance<=5 (Pag.2)	✓	✓	✓	✓	✓
Leitner et al. [11]	SLA Cost = Penalties (Pag.3)	✓	✓	✓	✓	✓
Buco et al. [4]	Penalties (Pag.12)	✓	✓	✓	n/a	n/a
	Penalties & Reward (Pag.18)	✗	✗	✗	n/a	n/a
Grabarnik et al. [7]	Penalties & Reward (Pag.7)	✓	✓	✓	n/a	n/a
Rana et al. [15]	ExecutionTime (Pages 6-7)	✓	✗	✗	n/a	n/a

Other examples taken from relevant related works are the following: Buco et al. propose in [4] an SLA management system, called SAM that provides penalties in a Service Level Management process. In the first studied example (page 12) the compensation function relates some penalties with a service property denoting the alert time for SLA managers. In the second example some penalties and rewards are specified depending on the service level fulfillment of the SLAs; these last example define overlapping values for the rewards (i.e. same value can have different compensations) and therefore, the compensation function cannot be defined. Grabarnik et al. propose in [7] a model that can be used to reduce total service costs of IT service providers using alternative delivery teams and external service providers. The studied example (page 7) includes penalties and rewards for a service property that represents the process execution time. Rana et al. identifies in [15] how SLOs may be impacted by the choice of specific

penalty clauses. From such a work we have studied an example (pages 6-7) that relates the penalty of different levels of service execution time. It is important to note that the compensation function does not meet the saturation property and therefore it does not fulfill the validity property we defined in Section 4. In case there is no SLO explicit (Buco et al., Grabarnik et al. and Rana et al.) the validity and consistency of the compensable guarantee check is non applicable (n/a).

In our approach we leverage utility functions of service properties to analyze the consistency of compensations. In business studies, utility function models are also analyzed as they are strongly dependent on customer preferences and behaviour. [3] describes a business scenario with cost, customer expectations and reputation variables where reward function follows a non-monotonic behaviour (based on satisfying preferences from different customers). Similarly, Fenghui Ren et al. analyze in [16] how utility function is obtained from customer objective function (i.e., customers timetable preferences affect how transactions distribute through commercial opening hours). In temporal-aware web services matchmaking where the QoS varies on specific time periods, utility function models are also leveraged to find the optimal service offer for a given demand with validity periods [12]. In [13] we proposed how to specify utility functions and time periods within the WS-Agreement specification [1]. Finally, utility functions are also composed and integrated within comprehensive preference models enabling the combination of several ranking mechanisms [6].

Angelov et al. propose in [2] a formal representation for contracts to detect and solve different kinds of conflicts. Although the proposed contracts representation supports penalties and rewards by means of reparation clauses, they are not validated against utility functions as proposed in the current paper.

6 Conclusions and Future Work

In this paper we characterize the concept of Compensable SLA that includes a definition of guarantees with penalties and rewards for involved parties when the Service Level Objective (SLO) is underfulfilled or overfulfilled. We motivate our proposal upon the study of three real world SLAs ranging from human-driven services to computing services and define a model to formally express Compensable SLAs by conceptualizing the Compensation Function as the appropriate artifact to consistently combine penalties and rewards in the context of a SLO. In order to support the modeling and analysis of Compensable SLAs, we extend the notion of validity and consistence presented in [14] with a formal definition of properties.

Moreover, we validate our formalization by modeling our motivational examples and check the different properties defined in order to analyse them. It is important to highlight that based on this study we have identified some inconsistencies that could be derived from typical human mistakes when using natural language in the definition of compensable SLAs.

Since a thorough study of the different kinds of real-world SLAs has not been performed, as future work we will analyse other real-world examples that

incorporate more complex situations such as guarantees defined over multiple service properties and non-monotonic utility functions. Examples of this kind of utility functions can be found in other business domains such as logistics, where delivery normally refers to a just-in-time situation so the penalty would be defined not only for a late delivery of the product, but also for an early delivery. Another line of future work will be to extend our analysis tooling framework IDEAS⁷ to incorporate to automatically check the different properties studied in this work, in order to detect singular situations and common pitfalls.

References

1. Andrieux, A., Czakowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement) gfd-r.192 (October 2011)
2. Angelov, K., Camilleri, J.J., Schneider, G.: A framework for conflict analysis of normative texts written in controlled natural language. *The Journal of Logic and Algebraic Programming* 82(5-7), 216–240 (2013)
3. Bar-Isaac, H., Deb, J.: What is a good reputation? career concerns with heterogeneous audiences. *International Journal of Industrial Organization* 34(0), 44–50 (2014)
4. Buco, M.J., Chang, R.N., Luan, L.Z., Ward, C., Wolf, J.L., Yu, P.S.: Utility computing sla management based upon business objectives. *IBM Systems Journal* 43(1), 159–178 (2004)
5. del Río-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortés, A.: On the Definition and Design-time Analysis of Process Performance Indicators. *Information Systems* 38(4), 470–490 (2012)
6. García, J.M., Junghans, M., Ruiz, D., Agarwal, S., Cortés, A.R.: Integrating semantic web services ranking mechanisms using a common preference model. *Knowl.-Based Syst.* 49, 22–36 (2013)
7. Grabarnik, G., Ludwig, H., Shwartz, L.: Management of service process qos in a service provider - service supplier environment. In: *The 9th IEEE Int. Conf. on Ent. Comp., E-Commerce, and E-Services (CEC/ECC)*, pp. 543–550 (July 2007)
8. International Working Group on Cloud Computing Resiliency (IWGCR). Downtime statistics of current cloud solutions (2012)
9. Kronz, A.: Managing of process key performance indicators as part of the aris methodology. In: *Corporate Performance Management: Aris in Practice*, pp. 31–44. Springer, Heidelberg (2006)
10. Leitner, P., Hummer, W., Dustdar, S.: Cost-based optimization of service compositions. *IEEE Transactions on Services Computing* 6(2), 239–251 (2013)
11. Leitner, P., Hummer, W., Satzger, B., Inzinger, C., Dustdar, S.: Cost-efficient and application sla-aware client side request scheduling in an infrastructure-as-a-service cloud. In: *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, pp. 213–220 (June 2012)
12. Martín-Díaz, O., Ruiz-Cortés, A., Durán, A., Müller, C.: An approach to temporal-aware procurement of web services. In: Benatallah, B., Casati, F., Traverso, P. (eds.) *ICSOC 2005*. LNCS, vol. 3826, pp. 170–184. Springer, Heidelberg (2005)

⁷ Available at <http://www.isa.us.es/IDEAS/>

13. Müller, C., Martín-Díaz, O., Ruiz-Cortés, A., Resinas, M., Fernández, P.: Improving temporal-awareness of WS-agreement. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 193–206. Springer, Heidelberg (2007)
14. Müller, C., Resinas, M., Ruiz-Cortés, A.: Automated analysis of conflicts in ws-agreement. IEEE Transactions on Services Computing, TSC (2013)
15. Rana, O.F., Warnier, M., Quillinan, T.B., Brazier, F., Cojocararu, D.: Managing violations in service level agreements. In: Grid Middleware and Services Chapter Title - Managing Violations in Service Level Agreements, pp. 349–358 (2008)
16. Ren, F., Zhang, M.: Bilateral single-issue negotiation model considering nonlinear utility and time constraint. Decision Support Systems 60(0), 29–38 (2014)

Steady Network Service Design for Seamless Inter-cloud VM Migration

Nihed Bahria El Asghar¹, Omar Cherkaoui², Mounir Frikha¹,
and Sami Tabbane¹

¹ University of Carthage, High School of Communications of Tunisia, Tunisia
`{nihed.bahria,m.frikha,sami.tabbane}@supcom.rnu.tn`

² University of Quebec in Montreal, UQAM, Canada
`cherkaoui.omar@uqam.ca`

Abstract. The ability to move virtual machines between physical hosts is a powerful tool that virtualization provides to guarantee an on demand, scalable and survivable networking such as the Cloud. This Virtual Machine (VM) migration can be handled seamlessly when a VM is moved between physical machines belonging to the same data center. However, seamless inter-Cloud VM migration (WAN migrations) cannot be guaranteed, mainly for today's delay-sensitive video streaming applications. We propose in this paper, a hybrid virtual network to transparently migrate large volume virtual machines between data centers. Our proposed network service is designed using control theory concepts. We propose also a Lyapunov-based controller to prove the hybrid migration network stability.

Keywords: Inter-Cloud, seamless migration, network stability, control theory.

1 Introduction

Cloud providers are deploying geographically distributed data centers (DCs) in order to "optimally" serve the needs of Cloud users around the world. In specific cases, such as data center failure, service discontinuity, servers overload, etc., the Clouds' federation is required. A federated Cloud computing environment (or *Inter-Cloud*) facilitates instantaneous, opportunistic, and scalable provisioning of application services, in order to achieve QoS targets under variable workloads, resources and network conditions. This can be done by "optimally" migrating Virtual Machines (VMs) between the Clouds. The optimal VM migration across the WAN has gained huge interest nowadays from the research community; Most of research studies focus on the optimal placement of the target VM in order to achieve energy saving [1], load balancing [2], or low latency by choosing the closest target data center. In our work, we are assuming that the target host of the VM is already known. Our major interest is to find the best *federation network service* that could achieve *seamless VM migration* between geographically distributed data centers. The first idea that comes to the mind to deal with this

interconnection problem is to follow the intra-Cloud connection network solution for VM migration.

Typical data centers architectures today consist of LANs of either two-level or three-level trees of switches. A three-tiered design has a core tier in the root of the tree, an aggregation tier in the middle and an edge tier at the leaves of the tree. A two-tiered design has only the core and the edge tiers [3]. Two recent data center networking proposals are *PortLand* and *VL2* architectures [4] based respectively on *Fat Tree* and *Clos* network topologies. The *PortLand* [5] proposal defines *PMAC* (Pseudo MAC) address during VM migration to differentiate between initial and target VMs. The target VM will so keep the MAC address of the initial VM and the migration will be transparent with regard to the application user. The *VL2* [6] proposal however, uses *Location specific addresses* (LA), changing during migration and *Application-specific addresses* (AA) which are location independent, for transparent VM migration. Both proposals are Layer2 solutions requiring to have all servers in the same Layer2 VLAN. To meet this requirement in a geographically distributed environment, the VLAN notion should be extended over Layer 3 networks. VPLS (*Virtual Private LAN Service*) [7] is an alternative solution to do this but it still has various limitations. To overcome them, Cisco has recently proposed an *Overlay Transport Virtualization* (OTV) *data center Interconnect* (DCI) solution [8] which can be thought of as MAC-address routing. In OTV, traffic destined for a particular MAC address is encapsulated in IP and carried through the IP Cloud to its MAC-address routing next hop. The rich information in the MAC-address routing protocol enables Layer 2 connectivity over Layer 3 networks based on MAC-address destinations [9].

Using one or other of these solutions to interconnect Clouds will be to establish virtual inter-data centers tunnels across the WAN. These tunnels may span multiple network technologies or what we call a *multi-layer network* [10]. Our object in this paper is to exploit the multi-layered substrate network characteristics to design a network service enabling seamless and transparent VM migration beyond the data center.

In section 2 of this paper, we will explain our network design solution for inter-Cloud VM migration based on the analysis of the migration process steps. Section 3 will be dedicated to the analytical formulation of the design optimization problem and in section 4, we will propose a controller design and prove that it can achieve the migration network convergence and stability.

2 Migration Network Design Idea

In this paper, we intend to extend the seamless VM migration in the Cloud-based data center to cover a distributed environment of data centers, with always no perceivable effect to the end user. Therefore, IT departments will be able to perform hardware maintenance, consolidate CPU and memory resources, or migrate mission-critical applications from a data center when necessary without affecting the service-level agreements (SLAs) of the applications.

A successful application migration heavily relies on the underlying network infrastructure. For that reason, it is extremely important that IP network be resilient, robust, and highly available... The IP network becomes binding when the applications have to be migrated across data centers. In fact, in this case, a large amount of data should be transparently migrated through the low bandwidth and high latency Wide Area Network [17]. Thus, in order to effectively design the inter-Cloud migration network, it is primary to focus on the migrated workload characteristics. These characteristics could be depicted when deeply analyzing the migration process.

2.1 VM Migration Strategies

VM migration refers to transferring *run-time data* of a VM from one physical machine (the source) to another physical machine (the target). After migration, the VM continues to run on the target machine. *Seamless migration* is a migration during which the VM seems to be running all the time from clients' perspective [11]. The efficiency parameters for VM migration are mainly the *downtime delay*, during which services are entirely unavailable because of the VM suspension, and the *total migration delay*, from the migration process beginning to the starting of the VM execution on the target machine. These parameters extensively rely on the amount and consistency of the migrated data.

Intra-data center solutions, either based on *XEN* ([13],[18]) or *VMware VMotion* ([19], [21]), use *shared disk storage* [22] in order to reduce the amount of data transferred and consequently reduce migration and downtime delays. In fact, with such shared disk storage solutions, we will not need to transfer the *whole-system state* of a VM, but we will focus only on the migration of memory and *run-time CPU states*. Such a solution cannot be applied to the inter-data centers VM migration since the access to a shared disk storage across the WAN will be costly and will increase the migration delay. For these reasons, we will assume in the rest of this paper that a VM has to migrate its whole-system state ([12], [13]) during its inter-Cloud mobility. Different migration strategies proposed in the literature can be applied to the whole-system state migration. The simpler is the *freeze-and-copy* strategy that freezes the VM to avoid file system consistency hazards, copy its whole-system state, and then restarts the VM at the destination [12]. Such a strategy leads to a severe downtime due to the large size of the storage data transferred [11]. The *post-copy strategy* ([13], [14]) first transfers memory and CPU state only, and delays the storage migration. It then fetches storage data on-demand over network. This post-copy strategy was proposed for intra-data center VM migration where the CPU state is first transferred and the memory transfer is delayed. Even if this method is used in a data center LAN, it will decrease system availability, for its dependency on two machines. Furthermore, the continuous access to the storage data is not feasible while dealing with long haul migrations (over the WAN).

The *pre-copy* migration is the most referenced strategy in the literature ([12], [13]). It consists of copying first the *storage state* to the target node while the VM continues to execute at the source node. During the migration all the write

accesses to the local storage are recorded and forwarded to the destination, to ensure consistency. After the VM resumes on the destination, all the write accesses must be blocked [12]. This method has been patented for inter-domain VM migration across optical networks [15]. Our network service design will be, so, based on this pre-copy strategy since it was already applied to an all-optical wide area network.

2.2 VM Migration Process Analysis

Figure 1 presents the flow diagram of the VM migration process over an optical WAN. We will focus firstly only on the steps involving network (steps 6 and 7). Upon receiving confirmation of the existence of a path between the source and destination computing systems (a lightpath in [15], step 4), the VMTC (*Virtual Machine Turnable Control*) issues a *migrate* command to the source computing system (step 5). In response, the source and destination computing systems engage in an iterative copy process for transferring a copy of the state of the VM to the destination computing system (step 6). When, at step 7, the iterative copy process converges to data synchronization, the VMTC reconfigures (step 8) the IP tunnel (VM-to-user IP tunnel), thereby seamlessly redirecting and connecting the client-side application to the VM, now executing on the destination computing system.

Y. Luo et al. [11] propose a three-phase whole-system live migration strategy (Fig. 2) for the network involving steps (6 and 7) of Fig. 1. During the first step, all the data of the virtual machine to migrate is saved in cache and copied to the target machine where the new VM is instantiated. Then, iteratively, the dirtied data since the previous iteration (saved in the cache) is sent to the target VM. These iterations are repeated until the dirtied data comes down a prefixed threshold. All this phase is a *pre-copy phase* during which the VM is still running on the initial computing system and so, there will not be any delay constraint during this phase. However, since the amount of transferred data is very high during this phase, it will require high bandwidth migration pipes. We define therefore this phase as a **bandwidth-sensitive phase**. The bandwidth-sensitive phase will work optimally if data pages can be copied to the target computing system faster than they are dirtied by the migrating virtual machine. But this will require very large bandwidth paths that the substrate network may not offer, or that may restrict the network availability.

The synchronization step begins by suspending the virtual machine execution on the source computing system. Then the remaining inconsistent disk and memory pages are sent to the target VM with the CPU state. The duration of this transfer is called the *downtime delay* since it corresponds to the suspension of the VM on the initial computing system. For transparent and live VM migration, this *stop-and-copy phase* should be executed at very low latency. We call it therefore, a **delay-sensitive migration phase**.

Our design idea for the migration network is therefore based on this partitioning of the VM migration process. We propose in [20] that the migration path will be:

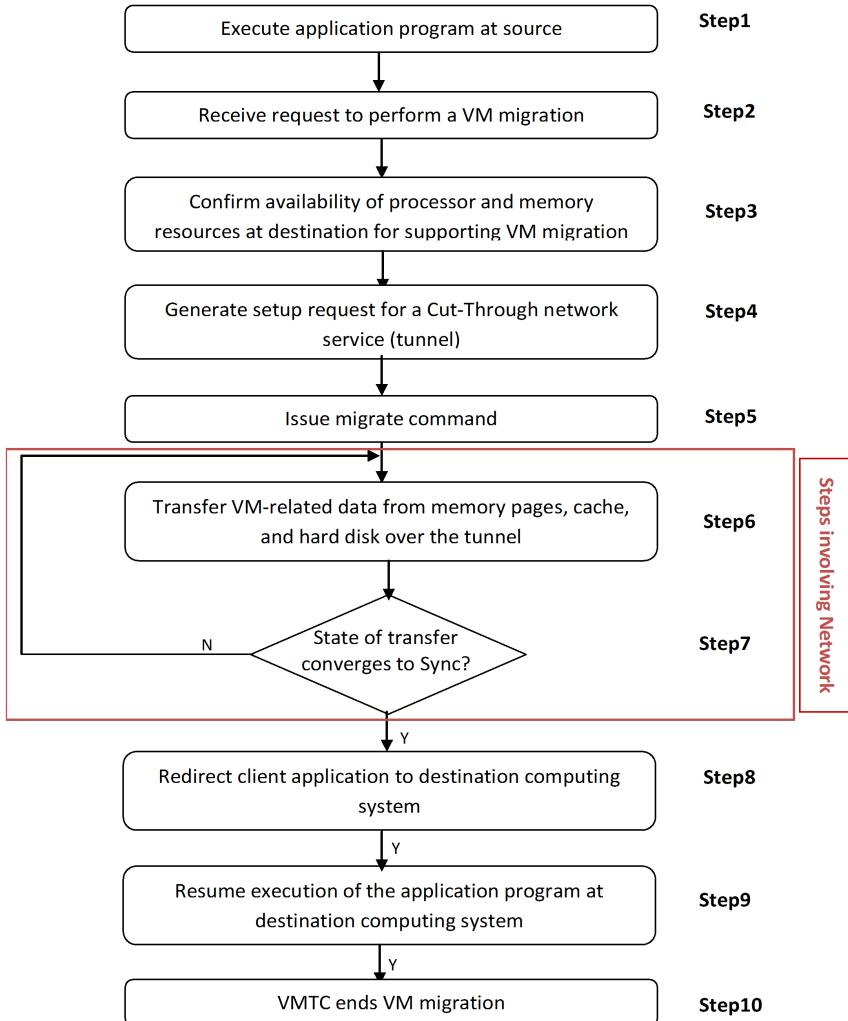


Fig. 1. VM migration process across an optical WAN [15]

- **A bandwidth-sensitive** virtual link during the iterative copy phase;
- **A delay-sensitive** virtual link during the stop-and-copy phase.

Our goal in this paper will be therefore to design the optimal migration network as a mix of a bandwidth-sensitive virtual network and a delay-sensitive VN. In the next section of the paper, we propose an analytical formulation of our design idea solution for the inter-Cloud VM migration as an optimization problem. We believe that our proposition will not only achieve live VM migration due to the delay-sensitive phase, but it will also reduce the total migration delay due to

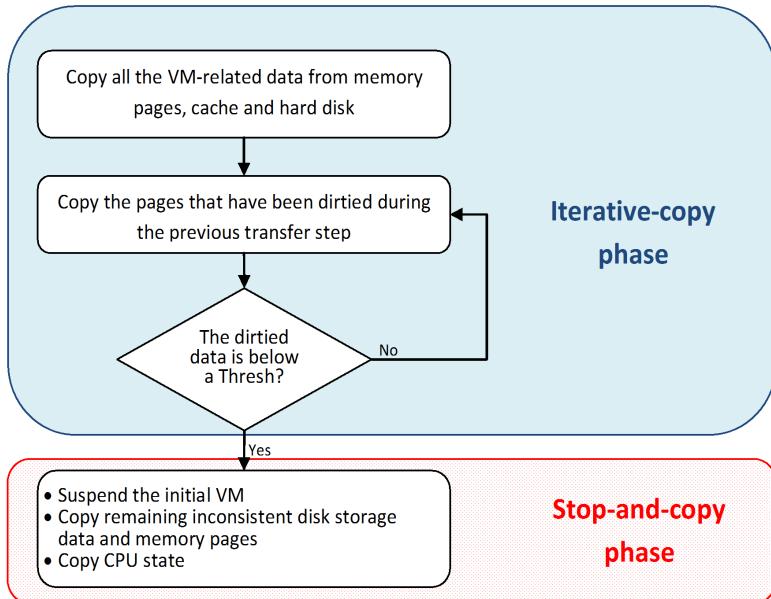


Fig. 2. Whole-system Virtual Machine migration process

the bandwidth-sensitive phase. In fact, the bandwidth-sensitive path can rapidly carry the migrated data which reduces the amount of dirtied data. Furthermore, the switching between two different-object migration paths will increase network resources availability.

3 Network Service Design Problem Formulation

Workload migration for inter-Cloud VM mobility is the principle concern for Clouds federation. We propose in this paper a network service providing on-demand inter-Clouds connectivity for the VMs mobility. Based on the multi-layered characteristic of the substrate wide area network supporting different switching paradigms in a single physical network, virtual networks are a promising solution for the design of an inter-Cloud network service. An optimal inter-cloud VN is a virtual network which can automatically adapt to the workload characteristics in terms of bandwidth and delay as depicted by the analysis of the migration process in the previous section. We propose to achieve this by designing a mix of two different-objective VNs for a better use of the substrate network and a better migration performance. The migration network service will be so a combination of a bandwidth-sensitive virtual network for the pre-copy phase and a delay-sensitive virtual network for the stop-and-copy phase.

The bandwidth-sensitive virtual network goal is to maximize, as much as possible, the offered bandwidth for each migration in the pre-copy phase in order to converge rapidly to the stop-and-copy phase. In fact when transferring memory pages on very high bandwidth links, there will not be numerous dirtied pages to copy during the following iteration, and they will rapidly reach the prefixed threshold to switch to the stop-and-copy phase.

The delay-sensitive virtual network is intended to reduce the stop-and-copy transfer delay in order to ensure the transparency of the migration with regard to the customer.

Such a multi-objective design has already been proposed in [16] where the authors propose a design model for multiple virtual networks, customized for different performance objectives in order to support diverse applications over a shared substrate. Similarly, we propose an optimization problem formulation for the design of two different objective virtual networks for carrying as much as possible migration requests. Let's denote $p = 1$ the index of the virtual network designed to serve the migrations in their bandwidth-sensitive network, and $p = 2$ the index of the virtual network designed to serve those migrations that are in their delay-sensitive phase.

The variables and parameters used in the formulation of the bi-objective network design problem are given below and summarized in table 1.

Problem Parameters

Let's a substrate network of L links connecting N hosts (that we assume belonging each to a different data center). Let c_l be the capacity of physical link $l \in L$, d_l be its length and v_l its propagation speed. Virtualization technology is deployed in order to allow on-the-fly instantiation of virtual machines on each host. Let K be the set of the deployed virtual machines. We say $k \in i$ if the virtual machine $k \in K$ is hosted on the physical machine $i \in N$.

We suppose that the dirtied pages to transfer each iteration are already calculated. Let's denote $(W^k \times \gamma_{ij}^k)$ the amount of data in VM k representing the pages to transfer from host i to host j . k will denote also the virtual link for VM k migration. $\gamma_{ij}^k = 1$ if the VM k is migrated from computing system i to computing system j , $= 0$ otherwise.

For each VM's k migration between computing systems i and j , the threshold on the transferred data rate h^k , from which the migration link will switch from the bandwidth-sensitive phase to the delay-sensitive phase is supposed to be known. This threshold will correspond to a change in the objective function for the corresponding migration. We define also the binary variable δ^k such that:

$$\delta^k = \begin{cases} 1 & \text{if } W^k > h^k \\ 0 & \text{else} \end{cases} \quad (1)$$

Table 1. Parameters and variables definition

Parameters	Definitions
(N, L)	physical network of N nodes and L links
c_l	capacity of physical link $l \in L$
d_l	length of physical link $l \in L$
v_l	propagation speed of physical link $l \in L$
(i, j)	source and destination hosts of a VM
K	set of virtual machines
k	$\in K$; index of a VM (and of its migration virtual link (i, j))
W^k	workload of VM k (unit: bytes)
γ_{ij}^k	= 1 if VM k is migrated from node i to node j
h^k	the threshold value of workload for VM k
δ^k	= 1 if we should use the bw-sensitive network ; = 0 if we should use the delay-sensitive network
Variables	Definitions
p	$\in \{1, 2\}$ migration phase index
M_l^{pk}	mapping variable of virtual link k on physical link l of network type p
b_p^k	bandwidth of virtual link k for virtual network p

Problem Variables

The migration network service can be determined by finding the optimal virtual link mapping and bandwidths assignments. We define so the following design variables:

$$\begin{aligned}
 - M_l^{pk} &= \begin{cases} 1 & \text{if } \text{virtual link } k \text{ is routed on} \\ & \text{physical link } l \text{ for network } p \\ 0 & \text{else} \end{cases} & p \in \{1, 2\} \\
 - b_p^k &: \text{bandwidth of virtual link } k \text{ for virtual network } p.
 \end{aligned}$$

Where M_l^{pk} is the mapping variable, and b_p^k is the bandwidth assignment variable.

Objective Function Formulation

According to the amount of data to migrate W^k (compared to the predefined threshold h^k), the migration is processed on a bandwidth-sensitive virtual link or a delay-sensitive virtual link. We define so two different objective functions:

1. Bandwidth-sensitive objective function

During the *bandwidth-sensitive phase* of the migration process (Fig. 2), a large volume of data should be transferred on high bandwidth links in order to reduce the dirtied data volume and converge rapidly to the threshold value. Our object in this part will be so to maximize the bandwidth of each virtual link, for all the admitted migration requests. This maximization should consider the amount of data to migrate on each virtual link and optimize the bandwidth allocation for all the simultaneous bandwidth-sensitive migrations. We will consider accordingly a *weighted proportional fair* [24] bandwidth allocation between the different virtual links. The weights are chosen to be the migrated data volume on the corresponding virtual links (the bandwidth-sensitive virtual links corresponding to $\gamma^k = 1$), in order to give more bandwidth for the virtual links that have more data to transfer.

$$U_1^k = \sum_{i,j=1}^N \gamma_{ij}^k \times \delta^k \times W^k \times \log b_1^k \quad (2)$$

2. Delay-sensitive objective function

During the *delay-sensitive phase* of the migration process, the main object is to reduce the downtime (period of inactivity of the VM for the data synchronization). Our object in this phase will be, so, to minimize the total delay (as the total sum of the propagation and transmission delays):

$$U_2^k = \sum_{i,j=1}^N \gamma_{ij}^k (1 - \delta^k) \left(\frac{W^k}{b_2^k} + \sum_l \frac{M_l^{2k} d_l}{v_l} \right) \quad (3)$$

– Objective function

Each virtual link k is established based on either the first bandwidth-sensitive objective or the second delay-sensitive function depending on the amount of data to transmit. The total objective function aims to optimize the routing and resource allocation for both virtual networks. It can be written as follow:

$$U(b, M) = \sum_{k \in K} U_1^k - \sum_{q \in K} U_2^q \quad (4)$$

Where $b = (b^k)$ ($b^k = \delta^k b_1^k + (1 - \delta^k) b_2^k$) is a vector of the bandwidths allocated to the different admitted migration requests, and M is the mapping matrix of elements $M_{k,l} = \delta^k M_l^{1,k} + (1 - \delta^k) M_l^{2,k}$. The added minus before the second part of the objective function is because we want to minimize the delay while the objective is a maximization function.

Optimization Problem Formulation

The optimization problem for the design of the bandwidth-sensitive and delay-sensitive virtual networks mix can be formulated as shown in equation 5.

$$\begin{cases}
\max_{b,M} U(b, M) \\
\text{subject to :} \\
\sum_{i,j} \gamma_{ij}^k \sum_{k,p} M_{mn}^{pk} b_p^k \leq c_{mn} \quad \forall (m,n) \\
\sum_m \sum_{ij} \gamma_{ij}^k M_{mq}^{pk} - \sum_n \sum_{ij} \gamma_{ij}^k M_{qn}^{pk} \begin{cases} = 0 & \text{if } q \neq i \& q \neq j \\ \geq 0 & \text{if } q = j \\ \leq 0 & \text{if } q = i \end{cases} \quad \forall k, q; \forall p = \{1, 2\} \\
M_{mn}^{pk} \in \{0, 1\}
\end{cases} \quad (5)$$

The first constraint is a physical link capacity constraint for link $l = (m, n)$. The second one is a mapping constraint ensuring path continuity and the final one is a value constraint.

The optimization model (5) presented above is designed to assign effectively the substrate resources to both the bandwidth-sensitive and delay-sensitive virtual networks for all the executing migration processes. Suppose that problem formulation (5) is solvable and let b_k^* and $(M_{k,l}^*)_{l \in L} \forall k$ denote the optimal values of the objective function for a prefixed value of the switching threshold h^k . This stationary solution requires the knowledge of all system parameters (like observable workload statistics and migrations' admission rate) in advance. It would not be adaptive to unpredictable changes in migrations' dirtied pages volume and must be recomputed. In the next section, we will present an online control algorithm that overcomes these challenges.

4 Optimal Network Control Algorithm

We focus here on an optimal control of the migration network mix by dynamically adjusting the switching threshold h^k in order to guarantee some tradeoff between the total migration delay and the downtime delay. In fact, for applications with pages dirtying rate exceeding the transmission rates of the migration links, the amount of data to be transferred will never reach the prefixed threshold value and the iterative copy phase will continue infinitely causing a high total migration delay. To overcome this, the network should dynamically adjust the threshold value in order to switch to the stop-and-copy (downtime) migration phase on the delay sensitive network.

4.1 Design of the Migration Network Controller

Let T_k be the duration spent to transfer a W^k bytes volume of dirtied pages over a b_1^k bps bandwidth-sensitive migration link. We note λ_k the dirtying rate of the

application executing on VM k . The amount of data to be transferred during the next iteration can be computed as follows:

$$W^k(t) = \lambda_k \times \frac{W^k(t - T_k)}{b^k(t - T_k)} \quad (6)$$

Equation 6 expresses that the dirtied volume to be transferred during the iterative pre-copy phase lies on the application's dirtying rate and the transfer rate on the migration virtual link.

Based on the above equation, we compute the dynamics of the transferred dirtied data volume as follows :

$$W^k(t) - W^k(t - T_k) = \left(\frac{\lambda_k}{b^k(t - T_k)} - 1 \right) W^k(t - T_k) \quad (7)$$

So:

$$\frac{W^k(t) - W^k(t - T_k)}{T_k} = \left(\frac{\lambda_k}{b^k(t - T_k)} - 1 \right) b^k(t - T_k) \quad (8)$$

Consequently, for very low transfer time, equation 9 can describe the dynamical behavior of the transferred volume, and eq. 10 describes the dynamical system's controller differential equation.

$$\dot{W}^k = \lambda_k - b^k(t - T_k) \quad (9)$$

$$\dot{W}^k = \lambda_k - \frac{W^k(t - T_k)}{T_k} \quad (10)$$

4.2 Optimal Control and System's Stability

In this section, we use the *framework of Lyapunov optimization* ([23], [24]) to develop a control algorithm that can be shown to achieve the optimal solution to the optimization problem (5) presented in section 3. The idea is to define a non-negative function, called a *Lyapunov function*, as a scalar measure of the aggregate workload of all the bandwidth-sensitive VM migrations in the network.

Specifically, we consider a virtual network of K VM migration pipes. Let $W(t) = (W^1(t), \dots, W^K(t))$ represent the vector of dirtied data volume in each VM to migrate as a function of time and \tilde{W} be the equilibrium point of the dynamical system (10). We define the following *quadratic Lyapunov function* $L(W)$:

$$L(W) = \sum_{k=1}^K (W^k - \tilde{W}^k)^2 \quad (11)$$

- L is a non-negative function and $L(\tilde{W}) = 0$.
- $\dot{L}(W(t)) = \sum_{k=1}^K \frac{\partial L}{\partial W^k} \times \frac{\partial W^k}{\partial t} = \sum_{k=1}^K 2W^k \dot{W}^k$

In order to ensure the Lyapunov asymptotic stability [24] of the system's equilibrium, \dot{W}^k should be strictly negative $\forall k$, since $W^k > 0$ and $\dot{L}\left(\tilde{W}(t)\right) = 0$.

The system stability condition is so given by Eq. 12 which indicates that a migration should be accepted by the bandwidth-sensitive virtual network if and only if the corresponding virtual link can handle a higher bandwidth than the data dirtying rate of the migrated VM.

$$\lambda_k < b^k (t - T_k) \quad (12)$$

Since the time when the stability condition is no more satisfied, the threshold h^k is updated in order to immediately switch to the delay-sensitive virtual network to execute the last migration step.

5 Conclusion

Cloud Computing is the expected solution to achieve the dream of computing as a utility. The ultimate wish is to use computing resources every where they are, even cross different Cloud data centers. In this context, some VMs could require inter-cloud migrations. This process should be transparent with regard to the VM's application user while migrated through a low bandwidth and high latency Wide Area Interconnection Network. We propose in this paper a network design for inter-cloud VM migration. The solution is based on a mix of a bandwidth-sensitive virtual network for the iterative phase of the migration process, and a delay-sensitive virtual network for the downtime phase. The network stability is based on the best choice of the migrated data threshold value to switch from the bandwidth-sensitive network to the delay-sensitive network. In this paper we propose a Lyapunov based proof of the migration network mix stability. As part of future work, we plan to support the analytical framework built in this paper with real system implementation to evaluate its performance.

References

1. Rajkumar, B., Rajiv, R., Rodrigo, N.C.: InterCloud UtilityOriented Federation of Cloud Computing Environments for Scaling of Application Services. In: Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), pp. 13–31 (2010)
2. Yawei, L., Zhiling, L.: A Novel Workload Migration Scheme for Heterogeneous Distributed computing. In: Proc. of 5th IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGrid 2005), vol. 2, pp. 1055–1062 (2005)
3. Al-Fares, M., Loukissas, A., Vahdat, A.: A Scalable, Commodity Data Center Network Architecture. SIGCOMM Computer Communication Review 38(4), 63–74 (2008)

4. Tavakoli, A., Casado, M., Koponen, T., Shenker, S.: Applying NOX to the Data-center. In: Proceedings of ACM HotNets (2009)
5. Mysore, R.N., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A.: PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In: Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, New York, NY, USA (2009)
6. Greenberg, A., Hamilton, J.R., Jain, N.: VL2: A scalable and flexible data center network. In: Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, New York, NY, USA (2009)
7. Juniper Networks: Implementing VPLS for Data Center Interconnectivity, Juniper Implementation Guide (2009), <http://www.juniper.net>
8. Cisco Systems: Technology Comparison: Cisco Overlay Transport Virtualization and Virtual Private LAN Service as Enablers of LAN Extensions. In: Cisco Systems white paper (2010)
9. Cisco Systems: Cisco Overlay Transport Virtualization, Data Center Interconnect Solution (2010), <http://www.cisco.com/>
10. Shiromoto, K., et al.: Requirements for GMPLS-Based Multi-Region and Multi-Layer Networks (MRN/MLN). In: IETF RFC 5212 (2008)
11. Luo, Y., Zhang, B., Wang, X., Wang, Z., Sun, Y.: Live and incremental whole-system migration of virtual machines using block-bitmap. In: Cluster 2008, pp. 99–106 (2008)
12. Bradford, R., Kotsovinos, E., Feldmann, A.: Live Wide-Area Migration of Virtual Machines Including Local Persistent State. In: ACM/Usenix International Conference on Virtual Execution Environments (VEE 2007), New York, NY, USA, pp. 169–179 (2007)
13. Hines, M.R., Gopalan, K.: Post-Copy Based Live Virtual Machine Migration Using Adaptive Pre-Paging and Dynamic Self-Ballooning. In: 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2009), New York, NY, USA, pp. 51–60 (2009)
14. Hines, M.R., Deshpande, U., Gopalan, K.: Post-Copy Live Migration of Virtual Machines. ACM SIGOPS Operating Systems Review 43(3), 14–26 (2009)
15. Travostino, F., Wang, P., Raghunath, S.: Seamless live migration of virtual machines across optical networks. United States Patent 7761573 (2010)
16. He, J., Zhang-Shen, R., Li, Y., Lee, C.-Y., Rexford, J., Chiang, M.: DaVinci: Dynamically Adaptive Virtual Networks for a Customized Internet. In: ACM CoNEXT 2008, vol. 136, pp. 1–12 (2008)
17. Cisco Systems: WAAS - Optimisation des applications. livre blanc, Cisco Systems (2006)
18. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live Migration of Virtual Machines. In: ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI 2005), vol. 2, pp. 273–286 (2005)
19. VMware: VMware VMotion: Live Migration for Virtual Machines Without Service Interruption. Product Datasheet (2009)
20. Bahria El Asghar, N., Cherkaoui, O., Frikha, M., Tabbane, S.: ICPN: An Inter-Cloud Polymorphic Network Proposal. In: Mellouk, A., Fowler, S., Hoceini, S., Daachi, B., Souih, S., Diaz, J. (eds.) WWIC 2014. LNCS, vol. 8458, pp. 243–256. Springer, Heidelberg (2014)

21. Cisco and VMware: Virtual Machine Mobility with VMware VMotion and Cisco Data Center Interconnect Technologies. In: Cisco white paper (2009)
22. Hansen, J.G., Jul, E.: Lithium: virtual machine storage for the cloud. In: Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC 2010), pp. 15–26. ACM, New York (2010)
23. Bastin, G.: Modélisation et analyse des systèmes dynamiques. In: Lectures notes, Louvain School of Engineering (2013)
24. Shakkottai, S., Srikant, R.: Network Optimization and Control. Foundations and Trends in Networking, now: the essence of knowledge 2(3), 271–379 (2007)

Prognosing the Compliance of Declarative Business Processes Using Event Trace Robustness

María Teresa Gómez-López¹, Luisa Parody¹, Rafael M. Gasca¹,
and Stefanie Rinderle-Ma²

¹ Language and Computer System Department, University of Seville
Avda. de la Reina Mercedes s/n, 41012, Sevilla, Spain
`{maytegomez,1parody,gasca}@us.es`

² University of Vienna, Faculty of Computer Science, Austria
Währingerstrasse 29, 1090 Wien, Austria
`stefanie.rinderle-ma@univie.ac.at`

Abstract. Several proposals have studied the compliance of execution of business process traces in accordance with a set of compliance rules. Unfortunately, the detection of a compliance violation (diagnosis) means that the observed events have already violated the compliance rules that describe the model. In turn, the detection of a compliance violation before its actual occurrence would prevent misbehaviour of the business processes. This functionality is referred to as proactive management of compliance violations in literature. However, existing approaches focus on the detection of inconsistencies between the compliance rules or monitoring process instances that are in a violable state. The notion of robustness could help us to prognosticate the occurrence of these inconsistent states in a premature way, and to detect, depending on the current execution state of the process instance, how “close” the execution is to a possible violation. On top of being able to possibly avoid violations, a robust trace is not sensitive to small changes. In this paper we propose the way to determine whether a process instance is robust against a set of compliance rules during its execution at runtime. Thanks to the use of constraint programming and the capacities of super solutions, a robust trace can be guaranteed.

Keywords: Declarative Business Process, Compliance Rules, Model-based Prognosis, Robustness.

1 Introduction

Monitoring the compliance of process instances with a set of compliance rules has been studied by several approaches, e.g., [1,2,3,4,5]. According to [3], the differences between reactive management and proactive management of compliance violations can be distinguished. On the one hand, reactive management means to diagnose the execution of a process instance for compliance violations.

On the other hand, proactive management refers to the prognosis of upcoming violations. In general, the main problem with diagnosis is that when the malfunction is detected, it is too late to avoid the non-compliance. Hence, when monitoring process compliance, prognosis of compliance violations before their actual occurrence seems to be promising. Existing approaches focus on the detection of inconsistencies between the compliance rules [1], or on monitoring process instances that are in a violable state [5].

In [4], a method to diagnose the inconsistencies between compliance rules of the model and the events produced by the process execution is proposed. That paper also includes a method to inform about the correct interval of time where the activities should have been occurred. Unfortunately, when these compliance violations are found, it means that the misbehaviour has occurred or is going to happen inevitably. In this paper, we propose how to avoid the misbehaviour by prognosing that an incorrect behaviour is close to happen, and by considering that we are on time to avoid the non-conformity. We extend and adapt the model-based prognosis using robustness to declarative business process monitoring. Model-based prognosis compares the expected compliance rules with the received event traces, including the analysis that small problems can occur in the future.

The advance knowledge of possible compliance violations empowers the execution to be more robust to malfunctions. *Robustness* can be understood as 'the ability of a system to cope with misbehaviours during execution'. On top of being able to avoid possible violations, a robust trace is not sensitive to small changes. The meaning of small changes depends on the type of problem, in our case it means possible delays in the execution of the activities. One way to solve the management of robustness found in the literature is by means of the use of (a-b)-super solutions [6]. We propose to adapt the key idea of the super solutions by means of applied it to compliance analysis in declarative business processes. We use the super solutions to know the possibility to find different alternatives of execution for the events in the future. This analysis helps us to be sure that, if a small unexpected behaviour occurs, we can find another consistent trace to the compliance rules. If it is not possible to find super solutions, it means that the trace is not robust to small changes, and an incorrect behaviour can occur in a more easy way. In order to achieve the prognosis of the compliance of declarative business processes in an automatic way, we propose in this paper:

- **Describe the Model-based Prognosis and the event trace model.** It implies to describe the declarative business process by means of compliance rules, and the observational model by means of an event trace. In order to describe the declarative business process, we use an extension of Declare [7] inspired in the Mobucon monitoring framework [8][9]. The proposed extension is derived from the example used in this paper, although the extension itself is not the aim of the paper.
- **Prognosticate the non-compliance using Event Trace Robutness and Find Critical Activities.** In order to determine a potential non-compliance of the business rules, the prognosis is performed. Using the robustness analysis of the event trace for a model, it is possible to detect a

non-conformity before it occurs. In order to perform it, we propose a computational model that verify, prognosticate, and determine the critical activities to maintain the conformity automatically. The proposal uses Constraint Programming paradigm. If a non-robust trace is found, since one of the activities can only be executed in an instant, then these activities need to be determined to advise the user about the importance to execute these activities in an instant, avoiding a non-conformity in the future.

The paper is structured as follows: Section 2 presents an example where a declarative model and a trace of events are shown, we have included what we expect from the prognosis analysis. Section 3 includes the necessary definitions to formalize and clarify the proposal. Section 4 analyses the necessary computational models to automatize the prognosis based on event trace robustness. Section 5 details the verification method, the prognosis process and the determination of the set of critical activities, using the approach explained in Section 4. In Section 6, the evaluation times and the complexity of resolutions are studied for an extended and real example. Section 7 analyses the previous related work found in the literature. Finally, conclusions and future works are presented.

2 Motivating Example

In order to motivate the importance to detect an inconsistency before it occurs, we use as example the protocol of pregnancy of the Health System of Andalusia. The example highlights the necessity to execute a set of activities in specific time intervals, and in accordance with other activities, due to some tests are related to the size of the fetus and/or the legal connotations of the gestation week. Although the whole example is evaluated in Section 6, only a fragment of the model is shown in Figure 1. To model the example, we have decided to used Mobucon with the time patterns extension of Declare, since it can fit the necessities of the example better than others. But we need to extent Mobucon framework, to model the interval of time where the activities can be executed and the time needed between the activities. The problem of modelling health protocols is an open problem in the area, as explained in [10]. The objective of our proposal is not based on the definition of a new language neither to extend the existing, only the establishment of the necessary patterns for the problem requirements.

Figure 1 represents nine activities where: one of them represents the initialization of the process (Amenorrhea); the activities execution order is described by means of precedence, response and succession relations; possible time interval constraints associated to the activities, and; maximum and minimum time distance between the execution of two activities. Also derived from the example, we assume that the execution of an activity is only represented by means of an occurrence in an instant (one event), not being necessary to represent or treat, the duration of the activities.

In order to understand the idea of prognosis, and its relation with the robustness, let us include the three possible reports and some examples of the

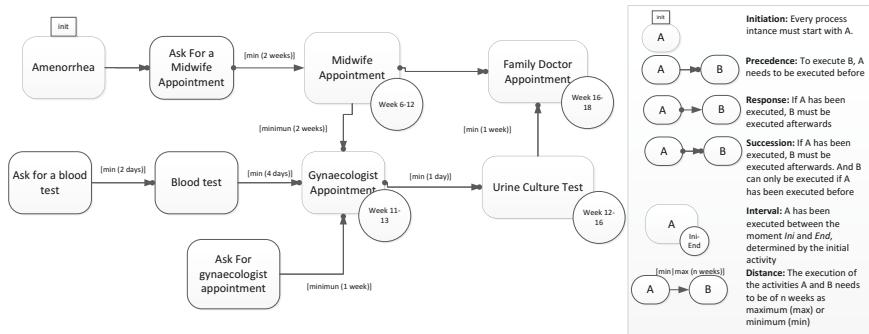


Fig. 1. Pregnancy example described by the Declare extension

corresponding traces, that our proposal can return during the monitoring, being each event described by means of the tuple $\langle \text{Time}, \text{Activity} \rangle$.

- **A correct and robust trace of events.** The trace $\{\langle \text{Day } 0, \text{Amenorrhea} \rangle, \langle \text{Day } 55 (\text{week } 8), \text{Ask for a midwife appointment} \rangle, \langle \text{Day } 69 (\text{week } 10), \text{Midwife appointment} \rangle, \langle \text{Day } 80 (\text{week } 12), \text{Ask for gynaecologist appointment} \rangle, \langle \text{Day } 81 (\text{week } 12) \}$, is correct and robust since every non executed activities can be executed in more than one instant in the future.
- **A correct although weak trace, since activities A_m, \dots, A_n can be only executed in one time instant in the future.** The trace $\{\langle \text{Day } 0, \text{Amenorrhea} \rangle, \langle \text{Day } 55 (\text{week } 8), \text{Ask for a midwife appointment} \rangle, \langle \text{Day } 69 (\text{week } 10), \text{Midwife appointment} \rangle, \langle \text{Day } 81 (\text{week } 12), \text{Ask for gynaecologist appointment} \rangle, \langle \text{Day } 81 (\text{week } 12), \text{Ask for blood test} \rangle, \langle \text{Day } 87 (\text{week } 13), \text{Blood test} \}$, is correct although weak, since activity *Gynaecologist Appointment* can only occur in the day 91, then if there is some problem that day, a non-conformity will occur.
- **An incorrect event trace where the events e_m, \dots, e_n are the responsible of the inconsistency.** The trace $\{\langle \text{Day } 0, \text{Amenorrhea} \rangle, \langle \text{Day } 55 (\text{week } 8), \text{Ask for a midwife appointment} \rangle, \langle \text{Day } 69 (\text{week } 10), \text{Midwife appointment} \rangle, \langle \text{Day } 81 (\text{week } 12), \text{Ask for gynaecologist appointment} \rangle, \langle \text{Day } 81 (\text{week } 12), \text{Ask for blood test} \rangle, \langle \text{Day } 87 (\text{week } 13), \text{Blood test} \rangle, \langle \text{Day } 92 (\text{week } 14), \text{Gynaecologist Appointment} \rangle \}$, is incorrect since the event $\langle \text{Day } 92 (\text{week } 14), \text{Gynaecologist Appointment} \rangle$ was executed too late.

3 Model-Based Prognosis in Declarative Business Processes

Compliance monitoring tends to be based on the comparison of the events that describe the execution of the process instances, and the compliance rules that define the policies to comply [3]. Unfortunately, the determination of a compliance

violation (diagnosis) means that the process execution has already produced an incorrect behaviour. In order to avoid a non-compliance, the prognosis detects signs of a possible misbehaviour before it occurs. We propose a computational model that warns about the critical activities during the execution of a process, to avoid the violation of the compliance in the future.

The architectures that support the monitoring has generally the following five modules: Process Execution (CRM, ERP, ...); Event Services; Compliance Monitoring; Compliance Requirements, and; Reporting and visualization of the compliance process. Since our proposal focuses on the prognosis of the non-compliance, the contribution is located in the Compliance Monitoring layer, where various activities are developed to support the prognosis of a declarative process model in function of the compliance requirements specified, and the events observed. The necessary activities to prognosticate (shown in Figure 2) are: Verification, Diagnosis, Prognosis based on Robustness, and Critical Activities Analysis. Each activity is related to the type of report that the proposal can offer: verify the correctness, diagnose an inconsistency, prognosticate a misbehaviour, or find the critical activities. Since how to verify and diagnose the non-compliance events was studied in a previous work [4], the two other tasks are faced in this paper, although to help in the prognosis, some improvements are included in the verification task.

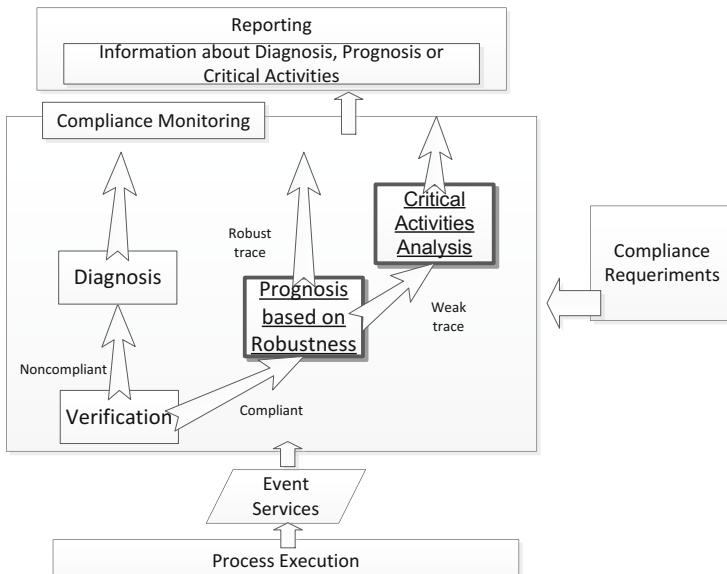


Fig. 2. Business Rule Monitoring Architecture

In order to describe the model that will be prognosticated, our approach adapts the definition of model-based diagnosis [11] to prognose declarative business processes.

Definition 1 (Prognosis Model (ProMod)). *It is formed by:*

- **Declarative BP Model (System Description):** *A declarative model is formed of a set of activities and a set of compliance rules. The compliance rules describe the relational order between the activities (precedence, response, succession), time intervals of execution associated to the activities ([Ini-End]), and time distance between the activities ([min | max (time)]).*
- **Event Trace (Observational Model).** *The events $\{e_1, \dots, e_p\}$ that make visible out of the process the execution of the activities between the initial time and the current time. Each event is associated to one and only one activity, although an activity can be executed several times and represented by several events.*

During the monitoring of a process execution, several event traces are observed. We assume that the events of these traces can be differentiated, being possible to analyse each trace separately. Each trace of events (partial or full) can be compliant or not, according to the compliance rules of the declarative BP model (Definition 1). A compliant partial instance can be *robust* or *weak*. In order to clarify all these terms, we include the following definitions:

Definition 2 (Full Trace (FT) and Partial Trace (PT)). *Being a trace A formed of the set of events $\{e_1, \dots, e_m\}$, where the set $\text{Executed}(A)$ represents the activities, without repetition, executed in the trace A. A is a Full Trace if there not exists a compliant trace B, such that $\text{Executed}(A) \subset \text{Executed}(B)$. In the opposite way, A is a Partial Trace iff there exists a compliant trace B such that $\text{Executed}(A) \subset \text{Executed}(B)$.*

Definition 3 (Promising Trace (PrT), Robust Trace (RT) and Weak Trace (WT)). *Being A a partial trace formed of the list of events $\{e_1, \dots, e_m\}$, A is a:*

- **Promising Trace,** *if A is a compliant PT for a Declarative BP Model.*
- **Robust Trace,** *if A is a PrT and, for all the non-executed activities of A according to its FT ($\text{Executed}(B) - \text{Executed}(A)$), there exists more than one instant to execute these non-executed activities. Formally, being: 'a' an activity; A a PrT, and; B and C, two FTs of A. A is a RT, iff $\forall a \in \{\text{Executed}(B) - \text{Executed}(A)\}$, there exists another FT C, where the only one difference between B and C is the execution time of the activity 'a' (t_B and t_C) respectively. The distance between the two execution times ($|t_B - t_C|$) depends on the smallest time reference included in the ProMod.*
- **Weak Trace,** *if A is a PrT although not a RT.*

4 Automating Robust-Based Prognosis Using Constraint Programming

In order to perform automatically the prognosis, and determine the critical activities of a declarative model, we propose the use of the concept of *super solution* of Constraint Programming. The use of Constraint Programming paradigm brings major advantages, such as early identification of non-compliance for a PT (even before all the activities are executed) [12]. In addition, several algorithms and tools have been developed, which can efficiently solve the model. In order to transform the ProMod into a constraint problem, it is necessary to explain: what is Constraint Programming, and how we can use it in model-based prognosis (Subsection 4.1); how to model the trace of events by means of constraints (Subsection 4.2), and; how to transform the declarative BP model into a CSP (Subsection 4.3).

4.1 Constraint Programming and Prognosis Using Robustness

With the aim of prognosticate possible inconsistencies using event trace robustness, we propose the use of the concept of super solutions of Constraint Programming [6]. An (a-b)-super solutions is a solution in which if the values assigned to a variables are no longer available, the solution can be repaired by assigning these a variables with new values and at most b modifications in other variables. To find a super solution guarantees the existence of a small set of repairs when the future changes in a small way. To be applied in prognosis, we use (1,0)-super solutions, that follows Definition 3 of RT and WT , where if one variable loses its value, we can find another solution by re-assigning this variable with a new value, without the modification of another variable.

To find the existence of (1-0)-super solutions in accordance to a PT , Constraint Satisfaction Problems (CSPs) can be applied.

CSPs represent a reasoning methodology consisting of a model a problem formed by variables, domains and constraints. Formally, it is defined as a triple $\langle X, D, C \rangle$ where $X = \{x_1, x_2, \dots, x_n\}$ is a finite set of variables, $D = \{d(x_1), d(x_2), \dots, d(x_n)\}$ is a set of domains of the values of the variables, and $C = \{C_1, C_2, \dots, C_m\}$ is a set of constraints. A constraint $C_i = (V_i, R_i)$ specifies the possible values of the variables in V simultaneously to satisfy R . Usually, to solve a CSP, a combination of search and consistency techniques is used [13].

When an objective function f has to be optimized (maximized or minimized), then a Constraint Optimization Problem (COP) is used, which is a CSP with an objective function f .

The specific CSP or COP created to automate the verification and the prognosis, will be detailed in Section 5. Before, it is necessary to know how to model the ProMod by means of constraints, to know after, how these constraints can be combined in a CSP.

4.2 Modelling Event Trace by Means of Constraints

As it was introduced in Definitions 2 and 3, to know if a *PT* is compliance or not, the activities that can be executed in the future (*FT*) need to be analysed. For this reason, it is necessary to distinguish between events that have already happened (observational model) and the events that will be thrown by the process execution (events of the future) [4]. For example, analysing the compliance rule that relate activities *Midwife Appointment* and *Family doctor Appointment*, if an event related to the activity *Midwife Appointment* is executed, *Family doctor Appointment* must be executed afterwards. The model needs to support the representation of the executed events, for example $\langle 52, \text{Midwife Appointment} \rangle$, and the events expected in the future, such as $\langle t_x, \text{Family doctor Appointment} \rangle$, where t_x represents an instant in the future ($t_x > \text{current Time}$) where the activity must occur to satisfy the model.

In order to capture the information on events of the past and the possibilities in the future, the model-based prognosis for each event (executed or not) must include a variable associated to the timestamp [4]: the pair of variables $\langle \text{Executed}, \text{Time} \rangle$ represents if the event has been executed with the Boolean variable *Executed*, and the instant when it was executed with the numerical variable *Time*. In particular, the following rules depending on the execution time of each event must be satisfied [4]:

- If the event e has been executed: $\text{Executed} = \text{true} \wedge \text{Initial Time} \leq \text{Time} \leq \text{Current Time}$.
- If the event e has not been executed but will be executed in the future: $\text{Executed} = \text{true} \wedge \text{Time} > \text{current Time}$.
- If the event e has not been executed and will not be executed in the future: $\text{Executed} = \text{false} \wedge \text{Time} = -1$.

4.3 Transforming Declarative BP Model into Numerical Constraints

In order to model the compliance rules of the *ProMod*, it is necessary to transform the patterns included in Figure 1 into computable constraints. Based on the model that describes the execution of the events that represents the activities, how the activity relations can be represented by means of numerical constraints is depicted in Figure 3. These transformations define, by means of numerical constraints, what each compliance rule means.

5 Verification of Compliance, Prognosis and Critical Activities Determination

The various activities related to model-based prognosis are based on Constraint Programming to determine if the event traces analysed during the monitoring are: *PrTs*, *RTs* or *WTs*. The example shown in Figure 4 represents a *PT*, where the various events located above the activities represent the execution of the activities of the declarative model. Thanks to the use of Constraint Programming,

Template	Logical Formula	Template	Logical Formula
	$A_{Ex} \wedge A_T = 0$		$A_{Ex} \rightarrow B_{Ex} \wedge A_T < B_T$ $\wedge B_{Ex} \rightarrow A_{Ex} \wedge A_T < B_T$
	$B_{Ex} \rightarrow A_{Ex} \wedge A_T < B_T$		$A_{Ex} \rightarrow A_T \geq \text{Ini} \wedge A_T \leq \text{End}$
	$A_{Ex} \rightarrow B_{Ex} \wedge A_T < B_T$		min: $A_{Ex} \wedge B_{Ex} \rightarrow A_T + n \leq B_T$ max: $A_{Ex} \wedge B_{Ex} \rightarrow A_T + n \geq B_T$

Fig. 3. Patterns of transformation from declarative BP model to numerical constraints

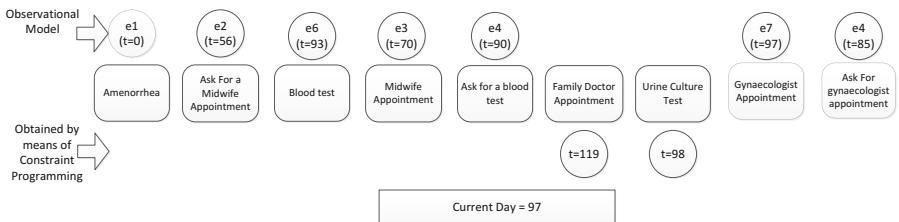


Fig. 4. Example of Trace

it is possible to know if this *PT* is a *PrT*. According to CSP, a *PT* is a *PrT* if it is possible to infer events for the non-executed activities (events below the activities), to obtain a *FT*. Following, how to ascertain if the event trace is a *PrT*, a *WT* or a *RT* is detailed.

5.1 Compliance Verification Using Constraint Programming

In order to compute the verification of a trace (*PrT*), we propose the creation of a correspondence between the elements of the *ProMod* and the parts of the CSP as follows:

- *Variables (X)* is formed by the variables that represent the execution and the instant of execution of the activities (A_{Ex} and A_T) for each activity of the model.

- *Domains* (D) is defined as Boolean for the variables A_{Ex} , and Integer for the variables A_T . A_T is represented by means of the smallest time reference included in the *ProMod*, days in our example. It is related to the quality of the event stream to be analyzed, in particular, the granularity of the recorded events [14]. Recording of events at a too coarse granularity might lead to imprecise or even incorrect results, e.g., in the medical domain [15]. The same holds for compliance monitoring and prognosis. In order to address this challenge, the proposed approach might be combined with techniques for enhancing event quality as proposed, for example, in [15].
- *Constraints* (C) is the set of compliance rules represented by means of numerical constraints, defined over the variables A_{Ex} and A_T , and derived from the transformation explained in Figure 3. Moreover, it is necessary to assign: the rules that relate the values between each pair of variables (A_{Ex} , A_T), as explained in Subsection 4.2; the values of the observed events to the variables A_{Ex} and A_T , and; the current time of the system to differentiate past and future events.

If a tuple of values for the variables X is found in the domain D , where all the constraints C are satisfiable, then we can assure that the trace is a *PrT*. In the opposite way, a diagnosis process will be necessary to find the event or events responsible of the misbehaviour. Table 1 shows a fragment of the CSP for the trace of Figure 4. To solve the CSP implies to solve the robustness problem.

A singularity of the declarative models is the possible representation of deviations from the prescribed model. This deviation might happen due to several reasons, such as exceptional situations, for the example, it might be necessary to see the midwife twice during week 6 to 12 (see Figure 1). This results in the multiple occurrence of activities [16] or Multiple Instance Patterns¹. The prognosis approach could deal with multiple activity occurrence by updating the related CSP once the occurrence of a multiple activity instantiation is detected.

5.2 Prognosis Using Event Trace Robustness and the Critical Activities Determination

Definition 3 of *RT* is aligned with the (1-0)-super solution in the sense that, a *PrT* is a *RT*, iff there exists a *FT*, such that every non-executed activities can be executed in at least two different instants of time. Therefore, to ascertain if an event trace is a *RT* or *WT*, it is necessary to find the *FT*, and the CSP of Table 1 needs to be modified to take into account various aspects:

- **How is the *FT* obtained?:** Since there are several *FT* that can satisfy a *PrT*, to perform the best prognosis, we have to take into account the widest *FT* for a *PrT*, given that more activities will be analysed. It can be obtained including an objective function that maximize the number of activities executed (the summatory of the A_{Ex} variables of the activities of the model) (as shown in Table 2). The objective function transforms the CSP into a COP.

¹ <http://www.workflowpatterns.com/patterns/control/>

Table 1. Example of Constraint Satisfaction Problem for Verification

```

//Variables (Time and execution for each activity) and Domains:
AmenorrheaEx, AskForMidwifeAppointmetEx, BloodTestEx, ...: Boolean
AmenorrheaT, AskForMidwifeAppointmetT, BloodTestT, ...: Integer
...
//Initialization of variables
currentTime = 97
initialTime = 0
//Compliance rules of the model
Amenorrheat = initialTime
AskForMidwifeAppointmetEx → AmenorrheaEx
    ∧ Amenorrheat < AskForMidwifeAppointmett
AskForMidwifeAppointmetEx → MidwifeAppointmentEx
    ∧ AskForMidwifeAppointmett < MidwifeAppointmentt
MidwifeAppointmentEx → FamilyDoctorAppointmentEx
    ∧ MidwifeAppointmentt < FamilyDoctorAppointmentt
    ∧
FamilyDoctorAppointmentEx → FamilyDoctorAppointmentEx
    ∧ MidwifeAppointmentt < FamilyDoctorAppointmentt
...
//Events occurred
AmenorrheaT = 0
AskForMidwifeAppointmetT = 56
MidwifeAppointmentT = 70
...
//Time Interval Execution
MidwifeAppointmentEx → 36 ≤ MidwifeAppointmentT ≤ 84
FamilyDoctorAppointmentEx → 106 ≤ FamilyDoctorAppointmentT ≤ 126
...
//Distance between activities
AskForMidwifeAppointmetEx ∧ MidwifeAppointmentEx →
    AskForMidwifeAppointmett +14 ≤ MidwifeAppointmentT
BloodTestEx ∧ GynaecologistAppointmentEx →
    BloodTestt + 4 ≤ GynaecologistAppointmentT
...
//Constraints to describe time and event execution relation
(AmenorrheaEx ∧ InitialTime ≤ AmenorrheaT ≤ currentTime) ∨ ...
(AskForMidwifeAppointmetEx ∧ InitialTime ≤ MarkedAsClearedT ≤ currentTime) ∨ ...
(MidwifeAppointmentEx ∧ InitialTime ≤ MidwifeAppointmentT ≤ currentTime) ∨ ...
...

```

Table 2. Finding the widest FT for the PT

```
//Objective Function
Maximize (AmenorrheaEx + ... + AskForMidwifeAppointmetEx)
```

- **Where (in which domain) is it possible to find another solution for a (1-0)-super solution?:** It will depend on the *rhythm* of the process, and the frequency of process monitoring: one per day in our example. But in general, it should be done after the current time, since it is not possible that an activity that has not occurred yet, takes an execution time of the past.
- **Which variables can 'lose' their values?:** In our case, the variables that can lose their values, according to the (1-0)-super solutions, will be the non-executed activities (*Executed(FT)-Executed(PrT)*), since the rest have already been executed in the past. The problem can appear when one activity executed in the future has only one day that satisfy the compliance rules, and finally this day occurs a problem, not being a robust solution. In order to know if a trace is a *RT*, we propose to include a variable in the CSP to represent that an activity can be executed in two different instants. The idea of the modification is shown in Table 3. If a solution is found, it is a (1-0)-super solution related to the variable a_t , since although this variable loses its value, it is possible to find another value a'_t that satisfy the compliance rules.

Table 3. Finding the widest FT for the PT

```
//Variables and Domain
at, bt, ct, a't
//Constraints
a+b=c
a'+b=c
at<a't ∧ currentTime < a',
```

An example of what it is necessary to include in the CSP of the example in Table 1, is the shown in Table 4, to analyse the robustness of *Urine Culture Test* in this case. If a tuple of values for *Urine Culture Test* is found, then it means that there is a (1-0)-super solution for the activity *Urine Culture Test*. In this CSP, it is also included the maximum number of activities to executed for a *FT*, value obtained in the COP of the verification process, nine in this example.

Since it is possible to find *WTs*, it is useful for the process execution to know the possible conflicts activities to be sure that they are executed in the unique instant that keep the compliance of the compliance rules. It means to inform about the variables that represents the events that imply the execution of the activities, that has no (1-0)-super solutions. The information of the Critical Activities is directly obtained from the evaluation of each CSP created as explained in Table 4.

Table 4. Example of Constraint Satisfaction Problem for Robustness Analysis

```

//Variables and Domains:
..., UrineCulturet, UrineCulturet': Integer
//Compliance rules of the model
...
FamilyDoctorAppointmentEx → UrineCultureEx
    ∧ UrineCulturet < FamilyDoctorAppointmentt
FamilyDoctorAppointmentEx → UrineCultureEx
    ∧ UrineCulturet' < FamilyDoctorAppointmentt
GynaecologistAppointmentEx → UrineCultureEx
    ∧ GynaecologistAppointmentt < UrineCulturet
∧
UrineCultureEx → GynaecologistAppointmentEx
    ∧ GynaecologistAppointmentt < UrineCulturet
GynaecologistAppointmentEx → UrineCultureEx
    ∧ GynaecologistAppointmentt < UrineCulturet'
∧
UrineCultureEx' → GynaecologistAppointmentEx
    ∧ GynaecologistAppointmentt < UrineCulturet'
//Time Interval Execution
...
UrineCultureEx → 78 ≤ UrineCultureT ≤ 112
UrineCultureEx → 78 ≤ UrineCultureT' ≤ 112
//Distance between activities
...
GynaecologistAppointmentEx ∧ UrineCultureEx →
    GynaecologistAppointmentt + 1 ≤ UrineCultureT
GynaecologistAppointmentEx ∧ UrineCultureEx →
    GynaecologistAppointmentt + 1 ≤ UrineCultureT'
UrineCultureEx ∧ FamilyDoctorAppointmentEx →
    UrineCulturet + 7 ≤ FamilyDoctorAppointmentT
UrineCultureEx ∧ FamilyDoctorAppointmentEx →
    UrineCulturet' + 7 ≤ FamilyDoctorAppointmentT
//Events occurred
...
//Constraints to describe time relation and event Execution
...
//To find (1-0)-super solutions
(UrineCultureT < UrineCultureT') ∧ (currentTime < UrineCultureT')
//For the FT
(AmenorrheaEx + AskForMidwifeAppointmetEx + BloodTestEx + ... = 9)

```

6 Evaluation of the Case Study

To analyse the applicability of our proposal, we have implemented the full example of pregnancy protocol. The whole model and some example of CSPs are available in <http://www.lsi.us.es/~quivir/index.php/Main/Prognosis>. It is formed of 28 activities, 33 relation orders (24 precedences, 7 responses, 2 successions), 12 interval relations, and 17 distance relations. In Figure 5 the execution times to (a) Verify the conformity of partial trace, and (b) Analyse the robustness of a partial trace, are shown. Each measurement of both graphics represents the Verification and Prognosis respectively, with the activities executed until the moment, for example: the first measurement represents that there is a *PT* formed of an event of *Amenorrhea* activity, the second measurement represents that there is a *PT* formed of two events of *Amenorrhea* and *Ask For Midwife Appoint...*, ... Both graphics show as soon as more variables are instantiated, the time decreases. This occurs since the number of variables to assign values and to verify is reduced in each test.

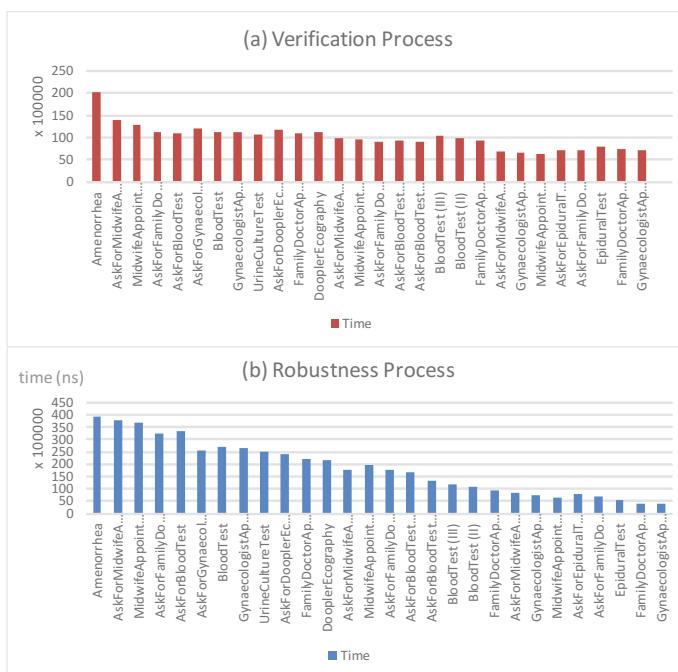


Fig. 5. Evaluation times for Verification and Robustness using Constraint Programming in the full example

7 Related Works

In this paper three challengers have been faced: how to model an example of a real medical guidelines; the use of the monitoring architecture, prognosis in our case, and; what technique can solve automatically the model-based prognosis. According to these challengers, the three corresponding areas have been studies.

Related to the languages of declarative models, several languages have been proposed to describe declaratively the ordering and temporal relations between activities in the business processes. Independently of the language, the common idea of declarative business process modelling is that a process is seen as a trajectory in a state space and that declarative constraints are used to define the valid movements in that state space [17]. The differences between declarative process languages are centred in the different perception of what is a state. Most relevant have been analysed in [18]. Some examples are the case handling paradigm [19], the constraint specification framework [20], the Declare language [21], PENELOPE language [22], Case Management (CMMN) [23], or EM-BrA²CE [24].

Related to the use of the monitoring architecture, it is possible to find works in literature related to verification, diagnosis, recommendations, There are many proposals that use the compliance rules and the monitoring of events to verify the correctness of a business process instance [25,1,5]. As discussed in [3], the *proactive management* of compliance violations during process runtime constitutes an important functionality. Even though the evaluation of existing approaches provided in [3] shows that some of them support proactive management, these approaches focus on detecting violations caused by the interplay between compliance constraints. In this approach, by contrast, the focus is on foresee violations that might become very likely in the course of process execution. This specifically necessitates to incorporate a notion of metric time as well as time distances and deadlines for process activities. Referring to [3] again, it can be argued that none of the existing approaches supports these temporal specifications within compliance constraints. Hence, the contribution of the proposed solution is two-fold: a) incorporation of time information that is presented in many real-world application (see the pregnancy example in this paper, but also the skin cancer treatment example provided in [26]); b) introducing and monitoring the likeliness of upcoming compliance violations (robustness or weakness of compliance for process instances).

Conforti et al. in [27] establish a recommended system in order to predict possible risks in the model. The event log is analysed in order to study the sequence of activities executed and the specific values consumed and provided by these activities. This analysis establishes the occurrence of possible faults (over-time, reputation-loss, and cost overrun), and provide a decision support for risk reduction. However, the proposal needs the historical information extracted from the event logs in order to provide a solution. At the same time, Maggi et al. in [28] also establish a predictive system in order to prevent customer to execute an instance which is not going to obtain the desired business goal. The authors establish a system to recommend which activity must be performed and

what data input values must the customer provide, but it is not treat the time restrictions used in medical guidelines.

About the techniques, the analysed languages use different knowledge representation paradigms, that enable different types of compliance rule management. For instance, Declare language is expressed in Linear Temporal Logic (LTL), whereas the PENELOPE language is expressed in terms of the Event Calculus.

Linear Temporal Logic (LTL) expressions can be used to represent desirable or undesirable patterns. LTL formula can be evaluated by obtaining an automaton that is equivalent to the formula and checking whether a path corresponds to the automaton. Unfortunately, the use of automatons does not allow to prognose according to time restrictions. It is due to our proposal does not only analyse the compliance rules activated because the antecedent was occurred. We include the whole model in the prognosis process as in [29], but with the difference that in this case a declarative language is used instead of an imperative language. On the other hand, the Event Calculus [30] is a first-order logic programming formalism that represents the time-varying nature of facts, the events that have taken place at given time points and the effect that these events reflect on the state of the system. Although one of the advantages of the use of event calculus is the ability to deductively reason about the effects of the occurrence of events and, more important is the abductive reasoning to discover a hypothesis about the malfunction to explain the evidence of events. But it does not have the capacity to propose a new set of data (events in this case) to avoid this misbehaviour, inferring possible misbehaviours in the future.

We have decided to use Constraint Programming since: it is a very mature area that has been applied to a wide range of problems, and with high level of complexity; it uses propagation techniques to reduce the search space in an efficient way; there are numerous tools and algorithms to model and solve problems, and; it permits an easy definition of the complex data using a wide range of constraints, such as implication constraints, disjunctive constraints, reified constraints, global constraints, and channelling constraints. Previous solution, as [29] [12], have used Constraint Programming to decision-making and diagnosis, respectively, over the input data of the process to avoid a failure, but not according to the execution moment of the activities, only for the input data of the activities, and in an imperative context.

8 Conclusions and Future Work

In this paper, we propose a framework to support model-based prognosis analysing the event trace robustness. According to the definition of robustness, we use the search of (1-0)-super solutions by using Constraint Programming. The (1-0)-super solutions can detect automatically the existence of critical activities, avoiding possible misbehaviours. It has been motivated by the use of a real example that represents the medical guidelines of the pregnancy protocol. The obtained evaluation times are very promising, thanks to use constraint programming.

As future work, we consider very interesting the analysis of: how the robustness can be related to the number of instances executed in each moment, since the systems have a limited number of resources, the distance between activities can be defined by the number of instances; how to facilitate the modification of the definition of robustness, to be implemented automatically, or; how it would be possible to define different degrees of robustness (low, medium, high, ...)

References

1. Maggi, F.M., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 132–147. Springer, Heidelberg (2011)
2. Montali, M., Maggi, F.M., Chesani, F., Mello, P., van der Aalst, W.M.P.: Monitoring business constraints with the event calculus. ACM TIST 5(1), 17 (2013)
3. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., Aalst, W.M.P.v.d.: A framework for the systematic comparison and evaluation of compliance monitoring approaches. In: 17th Int'l EDOC Conference (accepted for publication, 2013)
4. Gómez-López, M.T., Gasca, R.M., Rinderle-Ma, S.: Explaining the incorrect temporal events during business process monitoring by means of compliance rules and model-based diagnosis. In: EDOC Workshops, pp. 163–172 (2013)
5. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: Meersman, R., et al. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 82–99. Springer, Heidelberg (2011)
6. Hebrard, E., Hnich, B., Walsh, T.: Super solutions in constraint programming. In: Régin, J.-C., Rueher, M. (eds.) CPAIOR 2004. LNCS, vol. 3011, pp. 157–172. Springer, Heidelberg (2004)
7. Maggi, F.M., Westergaard, M., Montali, M., van der Aalst, W.M.P.: Runtime Verification of LTL-Based Declarative Process Models. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 131–146. Springer, Heidelberg (2012)
8. Maggi, F.M., Westergaard, M., Montali, M., van der Aalst, W.M.P.: Runtime verification of LTL-based declarative process models. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 131–146. Springer, Heidelberg (2012)
9. Maggi, F.M., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring business constraints with linear temporal logic: An approach based on colored automata. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 132–147. Springer, Heidelberg (2011)
10. Dunkl, R., Fröschl, K.A., Grossmann, W., Rinderle-Ma, S.: Assessing medical treatment compliance based on formal process modeling. In: Holzinger, A., Simonic, K.-M. (eds.) USAB 2011. LNCS, vol. 7058, pp. 533–546. Springer, Heidelberg (2011)
11. Struss, P.: New techniques in model-based diagnosis. In: Ramani, S., Anjaneyulu, K.S.R., Chandrasekar, R. (eds.) KBCS 1989. LNCS, vol. 444, pp. 428–437. Springer, Heidelberg (1990)
12. Gómez-López, M.T., Gasca, R.M., Pérez-Alvarez, J.M.: Compliance validation and diagnosis of business data constraints in business processes at runtime. Information Systems. Elsevier (2014)
13. Dechter, R.: Constraint Processing. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann (May 2003)

14. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. Lecture Notes in Business Information Processing, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
15. Binder, M., Dorda, W., Duftschmid, G., Dunkl, R., Fröschl, K.A., Gall, W., Grossmann, W., Harmankaya, K., Hronsky, M., Rinderle-Ma, S., Rinner, C., Weber, S.: On analyzing process compliance in skin cancer treatment: An experience report from the evidence-based medical compliance cluster (EBMC2). In: Int'l Conference on Advanced Information Systems Engineering, pp. 398–413 (2012)
16. Rinderle-Ma, S., Reichert, M., Jurisch, M.: On utilizing web service equivalence for supporting the composition life cycle. International Journal of Web Services Research (IJWSR) 8(1), 41–67 (2011)
17. Bider, I., Khomyakov, M., Pushchinsky, E.: Logic of change: Semantics of object systems with active relations. Autom. Softw. Eng. 7(1), 9–37 (2000)
18. Parody, L., Gómez-López, M.T., Gasca, R.M.: Data-oriented declarative language for optimizing business processes. In: International Conference on Information System Development, ISD 2013. Springer (2013)
19. van der Aalst, W.M., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. Data and Knowledge Engineering 53 (2005)
20. Sadiq, S.W., Orlowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. Inf. Syst. 30(5), 349–378 (2005)
21. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
22. Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 5–14. Springer, Heidelberg (2006)
23. OMG: Case management model and notation (cmmn). reference manual (2014)
24. Goedertier, S., Haesen, R., Vanthienen, J.: Em-bra²ce v0.1: A vocabulary and execution model for declarative business process modeling. FETEW Research Report KBI_0728, K.U.Leuven (2007)
25. Maggi, F.M., Montali, M., van der Aalst, W.M.P.: An operational decision support framework for monitoring business constraints. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 146–162. Springer, Heidelberg (2012)
26. Dunkl, R., Fröschl, K.A., Grossmann, W., Rinderle-Ma, S.: Assessing medical treatment compliance based on formal process modeling. In: Holzinger, A., Simonic, K.-M. (eds.) USAB 2011. LNCS, vol. 7058, pp. 533–546. Springer, Heidelberg (2011)
27. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 116–132. Springer, Heidelberg (2013)
28. Maggi, F.M., Francescomarino, C.D., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. CoRR abs/1312.4874 (2013)
29. Gómez-López, M.T., Gasca, R.M., Pérez-Alvarez, J.M.: Decision-making support for the correctness of input data at runtime in business processes. International Journal of Cooperative Information Systems 23 (2014)
30. Kowalski, R.A., Sergot, M.J.: A logic-based calculus of events. New Generation Comput. 4(1), 67–95 (1986)

Event-Based Real-Time Decomposed Conformance Analysis

Seppe K.L.M. vanden Broucke¹, Jorge Munoz-Gama², Josep Carmona²,
Bart Baesens^{1,3}, and Jan Vanthienen¹

¹ Department of Decision Sciences and Information Management, KU Leuven,
Naamsestraat 69, B-3000, Leuven, Belgium

² Universitat Politècnica de Catalunya, Barcelona, Spain

³ School of Management, University of Southampton, Highfield Southampton, SO17
1BJ, United Kingdom

{seppe.vandenbroucke,bart.baesens,jan.vanthienen}@kuleuven.be,
{jmunoz,jcarmona}@cs.upc.edu

Abstract. Process mining deals with the extraction of knowledge from event logs. One important task within this research field is denoted as conformance checking, which aims to diagnose deviations and discrepancies between modeled behavior and real-life, observed behavior. Conformance checking techniques still face some challenges, among which scalability, timeliness and traceability issues. In this paper, we propose a novel conformance analysis methodology to support the real-time monitoring of event-based data streams, which is shown to be more efficient than related approaches and able to localize deviations in a more fine-grained manner. Our developed approach can be directly applied in business process contexts where rapid reaction times are crucial; an exhaustive case example is provided to evidence the validity of the approach.

Keywords: real-time monitoring, process decomposition, conformance checking, conformance analysis, process mining, event logs.

1 Introduction

The research field of process mining deals with the extraction of knowledge from event logs, and has situated itself in the course of the past decade between the areas of Business Process Management (BPM) and data mining. As more and more process aware information systems are implemented, an increasing amount of event-based data is being recorded, which can hence be analyzed by process mining related techniques. An important task within process mining is called *conformance checking* (or, more broadly: *conformance analysis*), which aims to diagnose deviations and discrepancies between modeled behavior and real-life, observed behavior.

Conformance checking techniques face some hard and complex challenges in the context of today's organizations. First, the increasing amount of information

systems being implemented and applied to provide operational support, drive decisions and assist managers have led to a barrage of data, which should be parsed and analyzed in a manner which is both correct and scalable by conformance checking techniques. Second, given the current turbulent economic environment, stakeholders desire more than ever the timely delivery of reports and warnings, so that conformance checking techniques should no longer be applied in a post-hoc manner, after the actual occurrence of the activities being executed. Third, such techniques should be able to quickly and correctly localize and pinpoint deviating behavior and its root causes. As process models can become very complex, one wishes to highlight misbehaving parts in a running model, together with the ability to “zoom in and out” on these elements. Many conformance checking techniques have mainly been aiming to derive a global quality “metric”, denoting the global fitness or appropriateness of a process model, but without any real attention being applied towards localizing the main points of failure in an understandable manner.

In this paper, we propose a novel methodology to support real-time conformance analysis of event-based data streams, which aims to provide an answer to the challenges listed above. Our approach contributes to the current body of work in the following ways. First, we apply state of the art process model decomposition techniques [1] to split a large process model in a series of sub processes in order to gain a significant *speed-up* when verifying events. Second, by applying decomposition techniques, *localizing deviations* and volatile parts of the process models becomes more straightforward, allowing end-users to quickly gain an insight in which parts of the current model are failing or being violated. Third, by combining model decomposition techniques with a fast, event-granular replay technique, we are able to perform the conformance analysis task in a *real-time* manner, thus allowing for the monitoring of incoming events as they are being executed. This is a strong contribution compared to earlier approaches [2, 3, 4, 5, 6], where conformance checking techniques assume that a full recorded event log is available and where the actual analysis can be time-consuming.

The possible areas of application for our developed approach are manifold. In light of recent financial crises, the importance of the ability to immediately react to external shocks and unforeseen events has become more apparent than ever. Real-time monitoring, fraud detection and governance, risk and compliance (GRC) verification, and trading system failure protection all provide suitable contexts to apply our proposed technique. We apply our technique on a case example of a large bank transfer process to illustrate the validity of our contribution.

The remainder of the paper is structured as follows, Section 2 provides an overview of preliminary definitions and an overview of related work. Section 3 describes in detail our applied methodology, together with a technical description of the implemented artifact. Section 4 provides an empirical validation by describing a relevant case example and compares our technique against relevant approaches in this context. Section 5 concludes the paper and outlines opportunities for future work.

2 Preliminaries

This section provides an overview of preliminary definitions and concepts, as well as an overview of related work.

2.1 Related Work

Conformance checking techniques are devoted to quantify the quality of a process model in describing an event log. In the seminal work [7], a “fitness” metric is presented to describe the extent to which event traces can be associated with valid execution paths in the process model, and an “appropriateness” metric is proposed to assess whether the process model describes the observed behavior accurately enough. The aforementioned approach *replays* the traces of the log in the model to evaluate these metrics. One of the drawbacks of this approach is that for *undeterministic* models, the heuristics used in the replay may lead to overestimating the metrics, due to the artificial creation of superfluous tokens in the model. Several solutions have been proposed to overcome this issue. Weidlich et al. propose a system to check process model consistency based on “behavioral profiles” [8, 9]—which can be derived in a straightforward and efficient manner but with loss of some granularity regarding the exact traces which can be accepted by the model at hand. Adriansyah et al. propose an alternative approach where the concept of “alignments” are introduced in order to match an event trace with a path through the model as closely as possible [3, 5, 6].

Various *decomposition* approaches to improve process discovery and conformance checking tasks have been proposed. In [10], the notion of passages is used to decompose a process model and/or event log into smaller parts to speed-up process discovery and conformance checking. This approach has been generalized in [1] where it is shown that any event-granular process discovery and conformance checking tasks can be decomposed as long as the different process fragments (i.e. the submodels) only share uniquely-labeled activities. We apply this approach in this paper, but utilize a Refined Process Structure Tree (RPST) based decomposition method as outlined in [11, 12], as the hierarchical topological structure provided by this decomposition allows to enable additional analytical tasks not considered before, such as zooming in and out on various parts of the process model being monitored.

Our methodology also bears similarities with the fields of Complex Event Processing (CEP) [13, 14] and Business Activity Monitoring (BAM) [15, 16]. Although these methodologies also support the (near-)real-time monitoring of events, our approach differentiates itself in two ways. First, our approach stays in the general realm of process mining by starting from a process model and comparing this against a stream of incoming events which can be related to several running process instances. Second, as we apply a decomposition strategy over the given process model, this allows to immediately relate violations or discrepancies to specific areas within this model, thus improving the localization of the root-causes behind such deviations.

Our conformance analysis methodology is applicable on all process models on which event-granular semantics can be defined and which can be meaningfully decomposed into a series of submodels. We apply Petri nets throughout this paper as the representational language for prescriptive process models.

2.2 Definitions

Definition 1. (*Petri net, Workflow net.*) A Petri net [17] is a triplet $PN = (P, T, F)$ with P a finite set of places and T a finite set of transitions with $P \cap T = \emptyset$. F is the set of flows $F \subseteq (P \times T) \cup (T \times P)$. A place p is an input place of a transition t iff $(p, t) \in F$; similarly, p is an output place of t iff $(t, p) \in F$. The state of a Petri net is defined by its marking $M : P \rightarrow \mathbb{N}_0$. A transition is “enabled” in a given marking whenever all of its input places contain at least one token. Firing a transition then consumes a token from each input place and produces a token in each output place.

We also define the concepts of workflow graphs, system nets and full firing sequences.

Definition 2. (*Workflow graph, System net, Full firing sequence.*) Given a Petri net $PN = (P, T, F)$, the workflow graph is defined as the structural directed graph $G = (V, E)$ with $V = P \cup T$ and $E = F$. A system net is a triplet defined over a given Petri net $SN = (PN, m_i, m_o)$ where m_i and m_o define the initial and final markings of the Petri net, respectively. $(PN, m_1)[\sigma](PN, m_2)$ denotes that a sequence of transitions $\sigma \in T$ is enabled and can be fired starting from marking m_1 , resulting in marking m_2 .

Next, we provide the definition of an event log together with an event stream.

Definition 3. (*Event log, Event stream.*) Let event log L be defined as a multiset of traces (process instances) with the cardinality (or size) $|L|$ denoting the total number of traces in the log, including duplicates. A trace $\sigma^L \in L$ is a finite sequence of activity labels with length $|\sigma^L|$ and with σ_i^L the activity at position i in trace σ^L . The set of activities occurring in the event log is then denoted as $A = \{\sigma_i^L | \sigma^L \in L, i = 1 \dots |\sigma^L|\}$. For the purpose of our real-time conformance analysis methodology, we define an event stream $ES = \langle e_1, e_2, \dots \rangle$ as a sequence (finite or infinite) of arriving events with an event $e \in ES$ expressed as a tuple $(id, act, time)$, $act : ES \rightarrow A$ a function denoting the corresponding activity label for an event and function $id : ES \rightarrow \mathbb{N}$ denoting the case identifier. It is trivial to convert an event log L to an event stream ES if a global order relation can be established over the recorded activities in the event log.

To establish whether a given Petri net is able to correctly parse a given activity trace, a mapping between the transitions T in the Petri net and the activity alphabet A of the event log has to be established.

Definition 4. (*Mapping, Fitting trace*) Given a Petri net $PN = (P, T, F)$ and an event log L with activity alphabet A , let $\mu : T \mapsto A \cup (s, b)$ be defined as a

mapping between the transitions in the Petri net and activities in the event log, denoting the relation between a fired transition and the recorded label in the event log. Multiple transitions mapped to the same activity are denoted as *duplicate transitions*. Transitions can be mapped to a silent activity s (not observed in event log) and hence executed at-will whenever they are enabled. A trace σ^L “fits” a system $SN = (PN, m_i, m_o)$ when a full firing sequence $(PN, m_i)[\sigma](PN, m_o)$ can be found such that $\langle \mu(\sigma_i) | \mu(\sigma_i) \neq s, i = 1 \dots |\sigma| \rangle = \sigma^L$. Transitions can also be mapped to a non-silent, blocking activity b (also not observed in the event log), which can hence never be executed during fitting replay of an event log trace.

3 Methodology

This section presents the developed real-time decomposed conformance analysis approach. Fig. 1 provides a schematic overview of the approach, which can be split up in four phases, explained in the next subsections. The implemented prototype has been implemented as a collection of ProM plugins¹.

3.1 Phase 1: Decomposition

The first phase of the proposed methodology entails *decomposition*. Formally, the overall system net $SN = (PN, m_i, m_o)$ is broken down into a collection of subnets $\{SN^1, SN^2, \dots, SN^n\}$ such that the union of these subnets yields the original system net $SN = \bigcup_{1 \leq i \leq n} SN^i$. By means of decomposing the original model into a set of subnets we aim to achieve the following goals. First, fragment the conformance problems into a set of more comprehensive semantic elements aiding on the diagnosis. Second, restrict the possible pernicious effects of the heuristics decisions taken during the conformance analysis (see Phase 3 below). Third, speed-up the analysis compared with non-decomposed conformance checking techniques.

Due to the final goal of analyzing conformance, not all possible decomposition approaches are appropriate for this task. Only those *valid* decompositions that preserve the conformance integrity should be considered [1]. That is, given the original net and the decomposed version, the *original net perfectly conforms iff all the subnets in the decomposed setting perfectly conforms*. In other words, no conformance anomalies should be lost or introduced in the transition from the overall model to the decomposed one. In [1], the authors define a valid decomposition—applicable on Petri nets—as the decomposition that satisfies the following conditions:

1. Each arc of the overall net belongs to exactly one of the submodels, i.e., $F = \bigcup_{1 \leq i \leq n} F^i$ where $F^i \cap F^j = \emptyset$ for $1 \leq i < j \leq n$;
2. Each place of the overall net belongs to exactly one of the submodels, i.e., $P = \bigcup_{1 \leq i \leq n} P^i$ where $P^i \cap P^j = \emptyset$ for $1 \leq i < j \leq n$;

¹ ProM is an academic process mining framework to allow for rapid prototyping and plugin development. See: <http://www.processmining.org/prom/start>

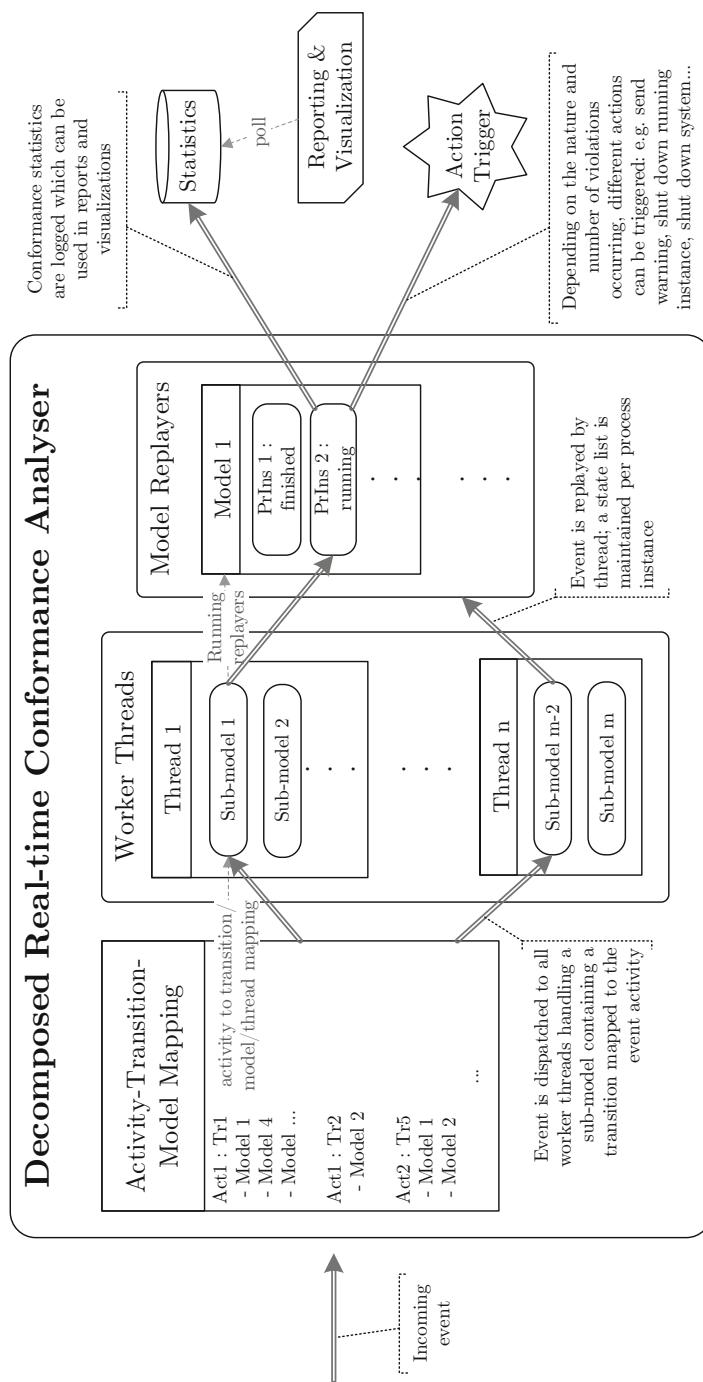


Fig. 1. Architectural overview of the developed real-time decomposed conformance analysis technique

3. Silent transitions appears in precisely one of the subnets, i.e., $\forall t \in T \setminus T_v(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| = 1$, where $T_v(SN)$ stands for the set of visible transitions (i.e., non-silent) of SN , i.e. $T_v(SN) = \{t \in T \mid \mu(t) \neq s\}$;
4. Non-silent, duplicate transitions appear in precisely one of the subnets, i.e., $\forall t \in T_v(SN) \setminus T_v^u(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| = 1$, where $T_v^u(SN)$ stands for the set of non-silent and non-duplicate transitions of SN , i.e. $T_v^u(SN) = \{t \in T \mid \mu(t) \neq s \wedge \nexists x \in T : x \neq t \wedge \mu(x) = \mu(t)\}$;
5. Non-silent, non-duplicate transitions may appear in multiple subnets, i.e., $\forall t \in T_v^u(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| \geq 1$.

In other words, all elements in the original Petri net model must belong to a submodel, but only non-silent, non-duplicate transitions can be shared among several submodels. In [1], the authors prove that any valid decomposition satisfying these conditions captures all the conformance problems of the overall model, and not more, i.e., it preserves the conformance integrity.

In [11], an approach based on SESE-components (Single-Entry Single-Exit) is presented, i.e. subgraphs in the workflow graph defined over a system net having single entry and exit boundary nodes [18]. The decomposition of the workflow graph of a Petri net into SESE-components is well-studied and provides a valid means to perform process model decomposition. In addition, SESE-components represent a well-defined and understandable part of the process model, with the added benefit that it is possible to define a hierarchical structure among the model fragments which allows to navigate through the different levels of granularity, so that a SESE-component perfectly reflects the idea of subprocesses within the main process. Fig. 2 depicts an example SESE-component for the illustrative case shown in Fig. 4, obtained using the technique proposed in [11]. Note that this techniques can be combined with a user-supervised post-processing step in order to obtain components that better fulfill the domain-aware monitoring.

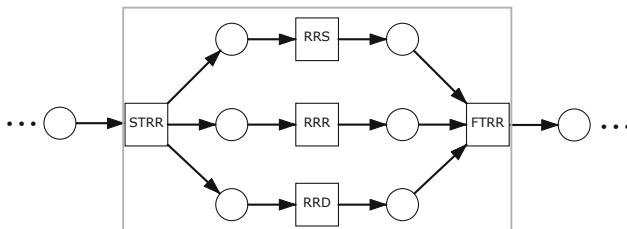


Fig. 2. “Open and register transaction” SESE-component from the case example in Fig. 5. *STRR* and *FTRR* are the *entry* and *exit boundary* nodes of the SESE-component, respectively. The rest of places and transitions are *interior* nodes of the SESE-component.

3.2 Phase 2: Event Dispatching

Once a system net has been decomposed into a set of submodels, this collection of models is passed to a central *event dispatcher*, which also serves to listen for incoming events. For each submodels, it is examined whether it contains a transition t which maps to the incoming event e , i.e. $\exists t \in T : \mu(t) = act(e)$. If it does, this indicates that the event at hand should be replayed on this particular submodel (multiple such submodels can be found), and the event is passed forward to this model fragment.

It is possible to decouple the “worker”-instances for each model fragment in a distributed fashion, with each model fragment running on separate machines. For the purpose of our prototype, we have implemented a multi-threaded architecture where a number of worker threads smaller than or equal to the number of process fragments is spawned, with each worker thread overseeing the handling of one or more process fragments. This approach allows for the concurrent handling of event checking over the different model fragments.

3.3 Phase 3: Replay

Once it is determined which process model fragment(s) should parse the incoming event, the actual *replay* of this event on each such fragment is performed. For each process model fragment, a state list is maintained denoting the current marking reached by the currently-running process instances. When an event e is queued for replay by a process fragment, the state linked to process instance $id(e)$ is progressed by investigating whether there exists an enabled transition $\exists t \in T : \mu(t) = act(e) \wedge enabled(t)$. The outcome of this evaluation determines if the process model is showing discrepancies or not.

Some additional remarks should be provided at this point. First of all, we note that we apply a heuristic, event-granular replayer similar to the one applied in [19]. The reasoning behind the choice to opt for a replayer playing the token game instead of an alternative approach such as alignment or behavioral profile based techniques [3, 9, 8] are twofold. First, alignment and behavioral profile based replayers perform their analysis on a trace, rather than event level, meaning that a complete process instance needs to finalize in order to align the log trace with a process model transition sequence. As we are dealing with event streams which need to be analyzed in a real-time manner, an event-granular replay strategy is required. Additionally, the behavioral profile approach does not specify how duplicate tasks can be dealt with. Second, alternative approaches suffer from scalability issues which make them unsuitable in a real-time context. Subsequently, the replay procedure applied here is formalized in Algorithm 1.

A second remark entails the way decision points are resolved by the replayer. Put briefly, whenever multiple (enabled) transitions are mapped to the same event log activity within a process model and/or whenever multiple invisible activities are enabled, the replayer needs to determine which transition to execute to handle the activity at hand. Note that—in extreme edge cases—it is possible that the forced firing of a non-enabled transition should be preferred if this avoids

several other violations later in the event trace [20]. In Algorithm 1, a replay strategy is put forward which prefers the firing of enabled transition mapped to the activity at hand first, followed by the set of silent transitions, followed by the set of non-enabled transition mapped to the activity at hand. If the chosen set contains multiple transition candidates, a one-step look-ahead procedure is executed to determine which candidate enables the execution of the following activity (if no such candidate can be found, a random one is chosen). For the multitude of process models, this look-ahead suffices to resolve any ambiguities. However, since we are dealing with streaming event data in this context, we possess no knowledge about events which will arrive in the future, preventing the execution of the look-ahead procedure. We propose and have implemented three methods to deal with this issue. First, disabling the look-ahead altogether and assuming that the model is deterministic enough to handle incoming events without taking the context into account ($n = 0$ in Algorithm 1). Second (another extreme), restarting the replay of the full trace each time an event is added, thus allowing the replayer to revise earlier decisions ($n = |\sigma^L|$ in Algorithm 1). Note however that the replayer is configured such that no new violations may be introduced related to historical activities ($\neg \text{conforms}(e) \vee \neg r$). In practice, this means that the replayer can revise the state chain by modifying the execution of silent transitions, selecting alternative albeit *also enabled* transition mapped to a particular activity for activities which were parsed correctly, or selecting alternative disabled transition, although only for activities which were not parsed correctly (provided by function *conforms* in Algorithm 1). The third method combines these two extremes by considering a part of the executed transition sequence as “frozen”, only allowing revisions for the last n steps.

As a third remark, recall that it was mentioned in Subsection 2.1 that one of the drawbacks of “token game”-based replayers entails the possible creation of superfluous tokens, enabling subsequently for too much behavior. However, as was mentioned in the description of Phase 1, we note that the decomposition of a process model restricts the possible pernicious effects of the heuristics decisions taken during the conformance analysis, as each model is now limited to dealing with a smaller subset of behavior. In addition, as superfluous tokens are created following the forced firing of violating activities, the process instance or model fragment at hand is likely to be immediately indicated as “dubious” at this point, lowering the trustfulness of following events within this instance of model fragment, independent of the replay strategy being applied. In addition, recall that we are applying a hierarchical decomposition strategy, so that it is possible to perform the actual replay at a lower-granularity level than the visualization and reporting.

3.4 Phase 4: Reporting and Visualization

The final phase consists of reporting and visualization. Remark that, naturally, these actions can be performed while the actual conformance analysis is running. In general, two ways of result follow-up are supported by our architecture. The first one consists of the logging of various statistics by the running worker threads

Algorithm 1. Real-time event replay algorithm.

Input: $PN = (P, T, F)$, $SN = (PN, m_i, m_o)$ % Given Petri net and System net

Input: $e = (id, act, time)$ % Arriving event to be replayed (checked)

Input: σ^L % Trace of log activities having occurred so far for instance $id(e)$

Input: σ % Trace of model transitions being executed so far for instance $id(e)$

Input: m % Current marking of the model for instance $id(e)$

Input: $conforms : \sigma^L \rightarrow \{\text{True}, \text{False}\}$ % Function denoting conf. outcome of previous event

Input: $enabled : (T \times M) \rightarrow \{\text{True}, \text{False}\}$ % Function denoting if a transition is enabled under a given marking (M is set of all possible markings)

Input: $nextmarking : (T \times M) \rightarrow M$ % Function returning the marking after (force) firing a given transition in a given marking

Input: $random : (T' \subseteq T) \rightarrow T$ % Function returning random transition from a given set of transitions

Input: $\mu : T \rightarrow A \cup (s, b)$ % Mapping function between model and log

Input: $n := 0$ % Number of steps to revise in historic trace (default: 0, i.e. none)

Input: $r := \text{False}$ % Denoting whether the event being replayed is a revised historic event

Input: $e_{next} := \emptyset$ % Next incoming event (optional; used when revising earlier decisions)

Output: Executed transition $t_{replayed}$

```

1: function REPLAYEVENT( $SN, e, e_{next}, \sigma^L, \sigma, m, n, r$ )
2:   % Handle revision of historic decision:
3:   if  $\neg r \wedge n > 0$  then
4:     Remove last  $n$  items from  $\sigma$  and revert marking  $m$  to earlier state
5:     for  $i \in \langle (n-1), \dots, 0 \rangle$  do  $ReplayEvent(SN, \sigma_{|\sigma^L|-i}^L, \sigma_{|\sigma^L|-i+1}^L, \sigma^L, \sigma, m, n, \text{True})$ 
6:   end if
7:   % Set up transition candidate collections:
8:    $EC^m := \{t \in T | enabled(t, m) \wedge \mu(t) = act(e)\}$ 
9:    $EC^a := \{t \in T | enabled(t, m)\}$ 
10:   $EC_f^m := \{t \in EC^m | \exists t' \in T : enabled(t', nextmarking(t, m)) \wedge \mu(t') = act(e_{next})\}$ 
11:   $EC_f^a := \{t \in EC^a | \exists t' \in T : enabled(t', nextmarking(t, m)) \wedge \mu(t') = act(e_{next})\}$ 
12:   $I_f := \{t \in T | enabled(t, m) \wedge \mu(t) = s \wedge$ 
       $\exists t' \in T : enabled(t', nextmarking(t, m)) \wedge \mu(t') = act(e)\}$ 
13:   $t_{replayed} := \emptyset$  % Transition chosen to be fired
14:  % Determine transition to fire:
15:  if  $|EC^m| > 0$  then % Transition is enabled
16:    if  $|EC_f^m| > 0$  then  $t_{replayed} := random(EC_f^m)$ 
17:    else  $t_{replayed} := random(EC^m)$ 
18:  else if  $|I_f| > 0$  then % Fire single invisible transition first
19:     $i' := random(I_f)$ 
20:    Update  $\sigma := \langle \sigma, i' \rangle$ 
21:    Update  $m := nextmarking(i', m)$ 
22:     $C := \{t \in T | enabled(t, m) \wedge \mu(t) = act(e)\}$ 
23:     $t_{replayed} := random(C)$ 
24:  else if  $|EC| > 0 \wedge (\neg conforms(e) \vee \neg r)$  then % Force fire
25:    if  $|EC_f| > 0$  then  $t_{replayed} := random(EC_f)$ 
26:    else  $t_{replayed} := random(EC)$ 
27:  else
28:    % No single transition is mapped to this event's activity, skip firing
29:  end if
30:  Update  $\sigma := \langle \sigma, t_{replayed} \rangle$ 
31:  if  $\neg r$  then Update  $\sigma^L := \langle \sigma^L, e \rangle$ 
32:  Update  $m := nextmarking(t_{replayed}, m)$ 
33:  return  $t_{replayed}$ 
34: end function

35: function REPLAYTRACE( $SN, \tau$ )
36:   % Function added for completeness, not used in real-time setting
37:    $m := m_i$ 
38:    $\sigma^L := \langle \rangle$ 
39:    $\sigma := \langle \rangle$ 
40:   for  $i \in 1..|\tau|$  do  $ReplayEvent(SN, \tau_i, \tau_{i+1}, \sigma^L, \sigma, m, 0, \text{False})$ 
41:   return  $\sigma$ 
42: end function

```

and replayers, which is polled regularly by decoupled components (e.g. a real-time dashboard or perhaps logged to a persistent data store). The second manner by which results can be interpreted consists of the definitions of various triggers which are to be fired once certain criteria are met, such as a model fragment overshooting a certain error rate threshold, for instance, of a high-risk activity or model fragment being violated. The actions which can be undertaken as a result are self-explanatory, e.g. sending warnings, or halting running process instances or even the complete system.

3.5 Implementation

Fig. 3 depicts a screen capture of our developed proof-of-concept implementation is shown. In the prototype, events are streamed over a network in real-time using a separate program (shown as the top left window in Fig. 3), which are received by the conformance analyzer and verified against the decomposed model fragments. The top panel displays a global overview of the model being checked against, with violating parts highlighted. Since the analysis is performed on the basis of the decomposed model fragments, it is more straightforward to pinpoint errors to a localized area within the global view than when using the full model as-is to perform conformance analysis. The lower left panels depict error monitors per submodel, showing the error rate for each model fragment over time. The panel on the right shows general statistics and program information. Note that this real-time approach allows to immediately react once a certain (user-configurable) criteria are triggered, such as model fragments (or specific activities) reaching a certain failure threshold., and shows the error rate for each model fragment together with a global overview for the complete process model.

4 Case Example

In this section we propose the study of a realistic process case example in order to illustrate the approach presented in this paper and its benefits. Model and logs—original and decomposed—of this case example, together with the rest of the benchmarks used in this experimental section, are publicly available².

4.1 Description

The modeled process describes a realistic transaction process within a banking context. The process contains all sort of monetary checks, authority notifications, and logging mechanisms responding to the new degree of responsibility and accountability that current economic environments demand. The process is structured as follows (Fig. 4 shows a high-level overview of the complete process): it is initiated when a new transaction is requested, opening a new instance in the system and registering all the components involved. The second step is

² doi:10.4121/uuid:c1d1fdbb-72df-470d-9315-d6f97e1d7c7c

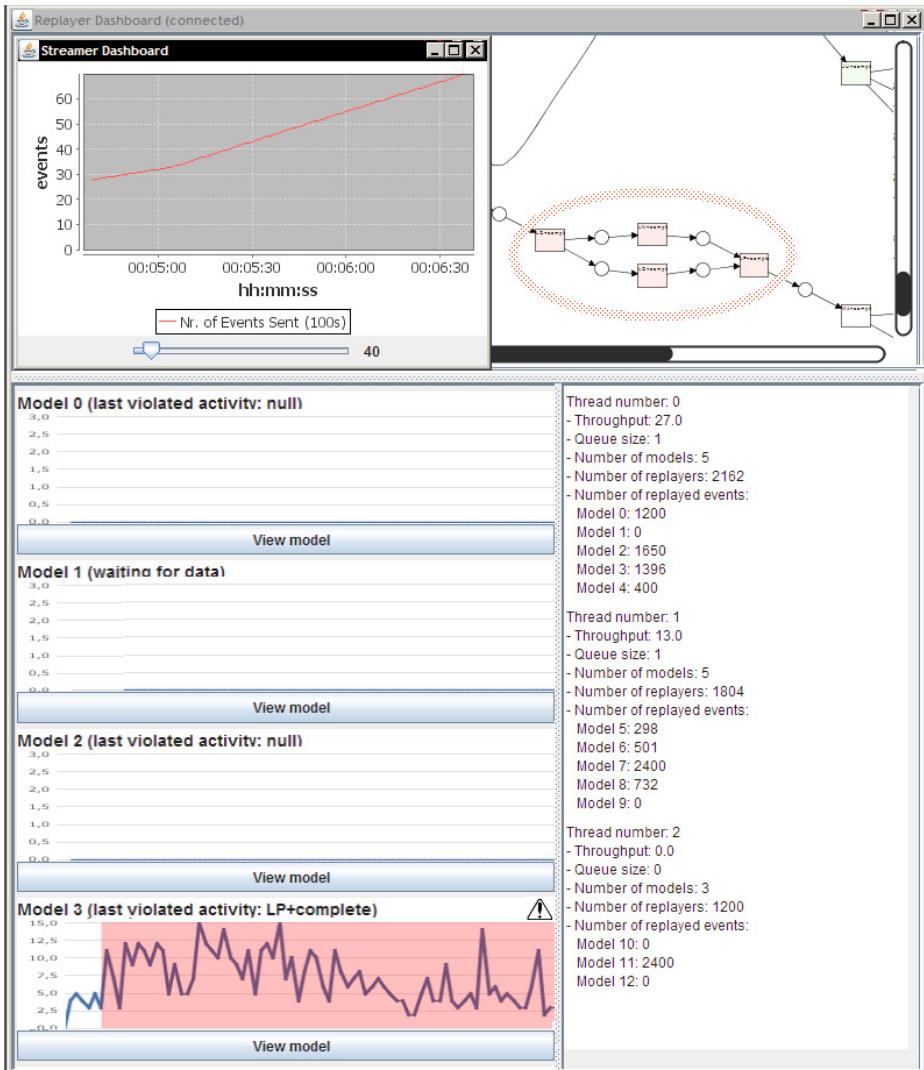


Fig. 3. Screen capture of the developed real-time event conformance analysis prototype. A global overview of the model being checked against, error rates per submodel, and general statistics are reported. Our real-time approach allows to immediately react once a certain (user-configurable) criteria are triggered, such as model fragments (or specific activities) reaching a certain failure threshold.

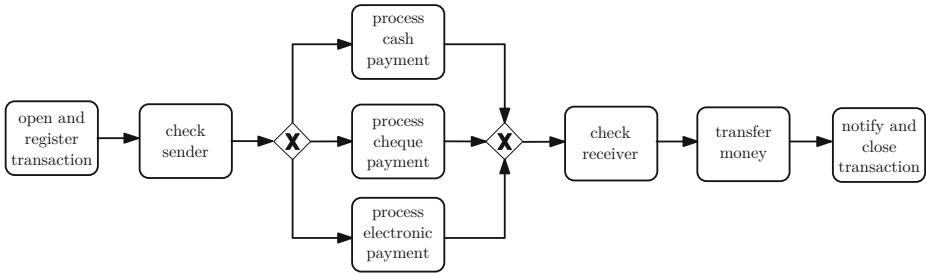


Fig. 4. High level overview of the running example process, structured in subprocesses

to run a check on the person (or entity) origin of the monetary transaction. Then, the actual payment is processed differently, depending of the payment modality chosen by the sender (cash, cheque³ and payment). Later, the receiver is checked and the money is transferred. Finally, the process ends registering the information, notifying it to the required actors and authorities, and emitting the corresponding receipt.

The process has been modeled in terms of a Petri net. The decomposition techniques based on SESE-components (see Section 3) is used to decompose the overall model into subprocesses. In particular, a valid decomposition where components have size at most 60 is derived. Finally, the decomposition is post-processed by merging some of the SESE-components in order to reach the final decomposition shown in Fig. 5 (which depicts the full process): eight of the proposed subnets correspond with the eight subprocesses identified in Fig. 4 (represented within gray rectangles), and the ninth subnet contains all the trivial connections between subprocesses (represented outside the rectangles).

4.2 Experimental Scenario Evaluation

To illustrate the benefits of the technique, we present two possible scenarios within the case example process.

Scenario 1: Serial Number Check. The modeled process defines that, whenever a client executes the payment in cash, the serial numbers must be checked (see Fig. 4). The banking regulation states that serial numbers must be compared with an external database governed by a recognized international authority (“Check Authority Serial Numbers CASN”). In addition, the bank of the case example decided to incorporate two complementary checks to its policy: an internal bank check (“Check Bank Serial Numbers CBSN”), and a check among the databases of the bank consortium this bank belongs to (“Check Inter-Bank Serial Numbers CIBSN”). At a given point, due to technical reasons (e.g., peak hour network congestion, malfunction of the software, deliberated blocking attack, etc.), the external check CASN is not longer performed, contradicting the

³ The British term is used to avoid ambiguity with the verb “to check”.

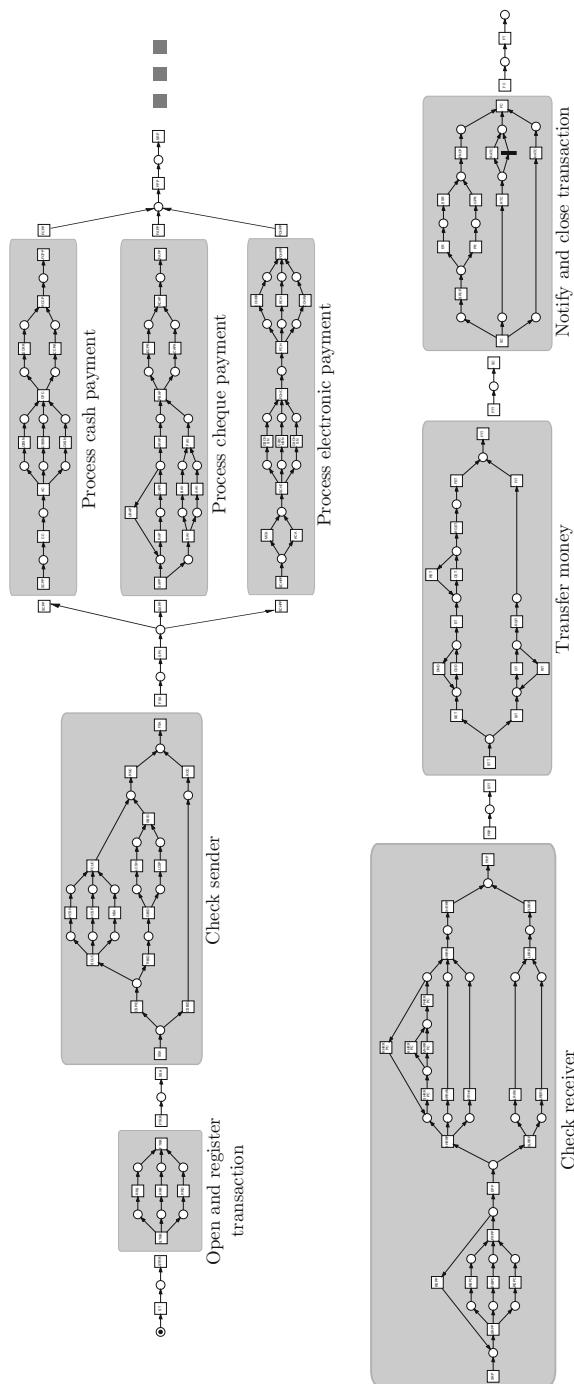


Fig. 5. Running example: final valid SESE-decomposition.
The substructures are named according to Fig. 4.

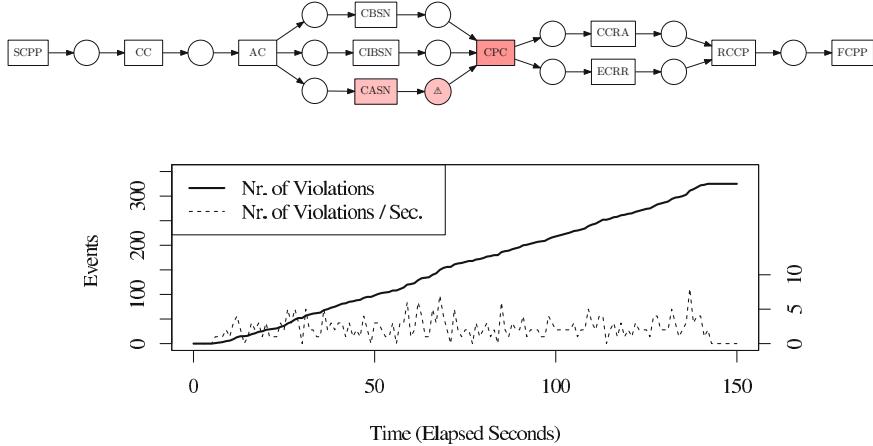


Fig. 6. In the first scenario, the *Check Authority Serial Number* (*CASN*) activity is skipped for some process instances, causing the *CPC* activity to fail, due to a missing input token which was expected to be present and placed there by the execution of *CASN*. The figure depicts the error localized in the affected model fragment; the graph depicts the cumulative and running amount of violations detected within this fragment.

modeled process, i.e., all the running instances of the process involving cash payment can proceed without the required check. Using the proposed approach, this situation is detected immediately, identifying the anomalous subprocess (*process cash payment*), focusing the conformance analysis on it, and eventually taking the necessary countermeasures. The consequences of detecting such cases only in forensic analysis performed months after the incident are severe and difficult to recover from. The situation is depicted in Fig. 6.

Scenario 2: Receiver Preliminary Profiling. During the *check receiver* stage, the model establishes two steps to be performed sequentially: first, a preliminary profiling analysis (“*Start Receiver Pre Profiling SRPP*”) is executed over the receiver in order to evaluate and establish its potential risk (“*Evaluate Pre Profiling EPP*”). Only then, a complete background check is performed over the receiver, where this check can either be more casual (“*Start Low Risk Receiver Processing SLRRP*”) or thoroughly (“*Start High Risk Receiver Processing SHRRP*”) depending on the potential risk detected on the preliminary profiling. However, the presence of an inexperienced bank employee, malevolence, or simply a bad implemented bank evaluation protocol, could result in evaluating the receiver with an unfinished preliminary profile check. The situation is depicted in Fig. 7.

4.3 Experimental Comparison

To benchmark the performance of our developed real-time conformance analysis technique against related approaches, a fitting event log was generated (based on

the model depicted in Fig. 5) containing ten thousand process instances (678864 events). A non-conforming (“noisy”) variant of this event log was produced by inducing noise (inserting, deleting, and swapping of events) so that 10% of the included events are erroneous.

We compare our proposed technique against the alignment based replay technique by Adriansyah et al. [3] as well our original implementation of the token-game based heuristic replayer [19]. Both the non-decomposed and decomposed variants of these techniques were included, applying hereto the methodology as described in [12].

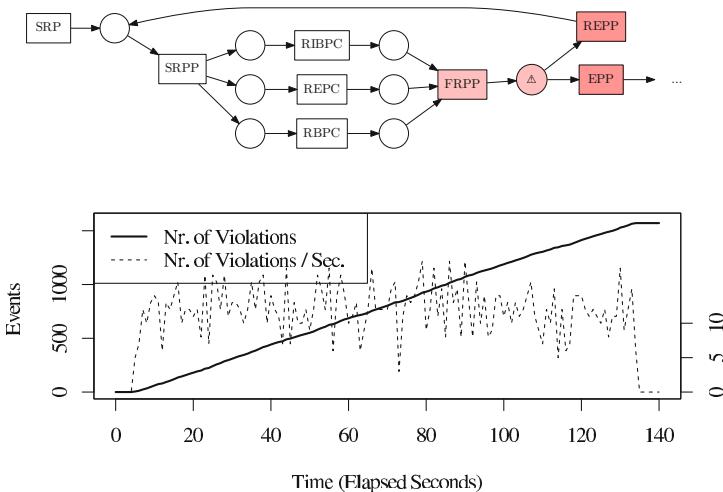


Fig. 7. In the second scenario, the preliminary profile check for receivers is skipped (*SRPP* to *FRPP*), causing either the *REPP* or *EPP* activities to fail. The figure depicts the error localized in the affected model fragment; the graph depicts the cumulative and running amount of violations detected within this fragment.

Fig. 8 depicts the performance results of the experiment, showing the amount of time taken (x-axis) to check the conformance of the included event logs (the y-axis represents the cumulative ratio of event checks performed). As can be seen, our proposed real-time conformance analysis technique performs competitively with respect to related techniques. During the experimental run, a maximum throughput rate (number of events checked per second) was reached at 35000 with the experiment running on a single consumer laptop with three worker threads. Some additional remarks should be provided however when interpreting Fig. 8. First, note that our proposed technique performs similarly as the heuristic decomposed replay technique, but note that our techniques executes a *conformance check on an event-granular basis* and thus can be applied in a real-time monitoring setting, whereas the other techniques do so on a trace-granular level (i.e. a complete trace should be provided to perform the replay procedure).

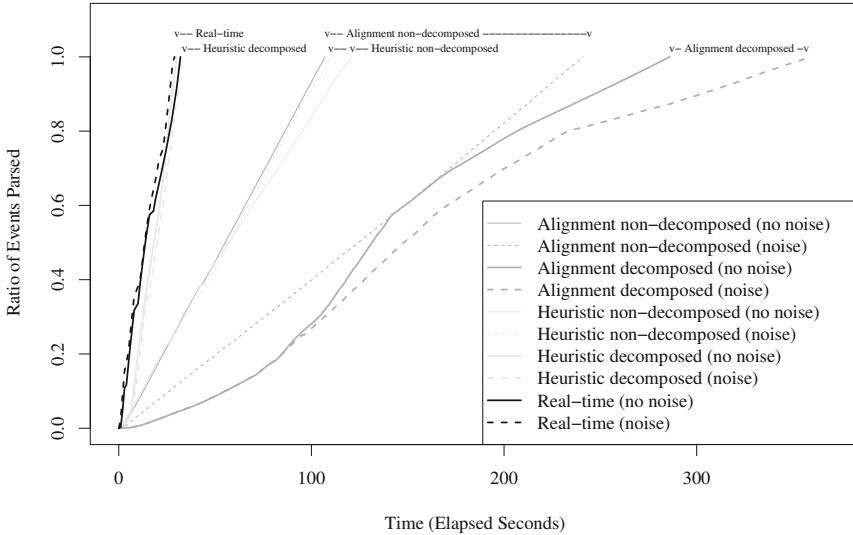


Fig. 8. Comparison of replay performance for the included techniques in the experimental setup, showing the time taken per technique to replay the given event log

However, the event log is of sufficient size so that a step-wise effect is not apparent in Fig. 8. Second, the replay procedure of the existing techniques was modified such that *each trace is checked independently* of the log context, meaning that no distinct trace grouping is performed over the log and each trace is checked as if it were belonging to an event log containing only this trace, so as to better assess the performance of these techniques in a real-time scenario (where the complete trace and log are unknown as events are arriving), rather than a post-hoc scenario where the complete event log is provided as-is. Note that—for the alignment based technique—this causes the non-decomposed version to perform better than the decomposed one. This is a perhaps unexpected result, but is caused by the fact that the alignment based techniques are geared towards checking—and as such expect—event logs as a whole. We thus emphasize the fact that these techniques have—currently—not been optimized to be applied in a real-time scenario (with an event stream being checked instead of an historical log).

5 Conclusions and Future Work

In this paper, we have presented a novel business process conformance analysis technique which is able to support real-time monitoring of event-based data streams. Our approach offers a number of novel contributions, most notably a speed-up compared to related techniques, the ability to localize discrepancies and allowing real-time monitoring and thus rapid response times in mission-critical

or high-risk environments, which is a significant benefit compared to existing conformance checking techniques which mainly work in an offline manner.

Future lines of research include: streamlining visualization and reporting capabilities of our prototype, incorporating other decomposition and replay strategies, and adapting the framework into a distributed implementation, where different replayer engines run on separate machines. In addition, we plan to pursue additional real-life case studies to confirm the validity of our approach.

Acknowledgments. This work is supported by the KU Leuven research council (grant OT/10/010), the Flemish Research Council (Odisseus grant B.0915.09), FPU grant (AP2009-4959) and project FORMALISM (TIN-2007-66523).

References

- [1] van der Aalst, W.M.P.: Decomposing Petri nets for process mining: A generic approach. *Distributed and Parallel Databases* 31(4), 471–507 (2013)
- [2] Adriansyah, A., Muñoz-Gama, J., Carmona, J.: Alignment Based Precision Checking. In: *BPM Center Report*, 12-10-2012 (2012)
- [3] 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)
- [4] Adriansyah, A., van Dongen, B.F.: Conformance checking using cost-based fitness analysis. In: *Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2011)*, pp. 55–64 (2011)
- [5] Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In: La Rosa, M., Soffer, P. (eds.) *BPM Workshops 2012. Lecture Notes in Business Information Processing*, vol. 132, pp. 137–149. Springer, Heidelberg (2013)
- [6] van der Aalst, W., Adriansyah, A.: Replaying history on process models for conformance checking and performance analysis. *WIREs Data Mining and Knowledge Discovery* 2, 1–18 (2012)
- [7] Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. *Information Systems* 33(1), 64–95 (2008)
- [8] Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. *IEEE Trans. Software Eng.* 37(3), 410–429 (2011)
- [9] Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on behavioral profiles. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) *ICSOB 2010. LNCS*, vol. 6470, pp. 1–16. Springer, Heidelberg (2010)
- [10] van der Aalst, W.M.P.: Decomposing process mining problems using passages. In: Haddad, S., Pomello, L. (eds.) *Petri NETS 2012. LNCS*, vol. 7347, pp. 72–91. Springer, Heidelberg (2012)
- [11] Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Conformance checking in the large: Partitioning and topology. In: Daniel, F., Wang, J., Weber, B. (eds.) *BPM 2013. LNCS*, vol. 8094, pp. 130–145. Springer, Heidelberg (2013)
- [12] Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Hierarchical conformance checking of process models based on event logs. In: Colom, J.-M., Desel, J. (eds.) *Petri NETS 2013. LNCS*, vol. 7927, pp. 291–310. Springer, Heidelberg (2013)

- [13] Gal, A., Hadar, E.: Generic architecture of complex event processing systems. In: Hinze, A., Buchmann, A.P. (eds.) *Principles and Applications of Distributed Event-Based Systems*, pp. 1–18. IGI Global (2010)
- [14] Zarri, G.P.: Representation and processing of complex events. In: AAAI Spring Symposium: Intelligent Event Processing, p. 101. AAAI (2009)
- [15] Kang, B., Lee, S.K., Bin Min, Y., Kang, S.H., Cho, N.W.: Real-time process quality control for business activity monitoring. In: Gavrilova, M.L., Gervasi, O., Taniar, D., Mun, Y., Iglesias, A. (eds.) *ICCSA Workshops*, pp. 237–242. IEEE Computer Society (2009)
- [16] Janiesch, C., Matzner, M., Müller, O.: Beyond process monitoring: a proof-of-concept of event-driven business activity management. *Business Proc. Manag. Journal* 18(4), 625–643 (2012)
- [17] Murata, T.: Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4), 541–580 (1989)
- [18] Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: Bravetti, M., Bultan, T. (eds.) *WS-FM 2010. LNCS*, vol. 6551, pp. 25–41. Springer, Heidelberg (2011)
- [19] vanden Broucke, S.K., Weerdt, J.D., Vanthienen, J., Baesens, B.: Determining process model precision and generalization with weighted artificial negative events. *IEEE Transactions on Knowledge and Data Engineering* 99(PrePrints), 1 (2013)
- [20] vanden Broucke, S., De Weerdt, J., Vanthienen, J., Baesens, B.: On replaying process execution traces containing positive and negative events. Feb research report kbi 1311, KU Leuven (2013)

Capitalizing the Designers' Experience for Improving Web API Selection

Devis Bianchini, Valeria De Antonellis, and Michele Melchiori

Dept. of Information Engineering University of Brescia

Via Branze, 38 - 25123 Brescia, Italy

{devis.bianchini, valeria.deantonellis, michele.melchiori}@unibs.it

Abstract. Solutions for supporting Web API selection may depend, behind the compliance of available Web APIs with respect to the search request (according to user-specified tags, Web API technical features, such as protocols or data formats, Web API categories), on votes assigned to Web APIs by web application designers who used them in the past. A web application designer, who is looking for a Web API for his/her own mashup, may learn from Web API rating performed by other designers. Votes should be properly weighted considering the experience/skill on application development gained by designers who assigned the votes. Therefore, estimating this experience is crucial for exploiting it in the best way. In this paper, we propose new techniques for the estimation of this experience by combining several factors, such as the reputation and popularity of the applications developed by the designers in the past. We validated our proposal with preliminary experiments, based on the contents of a well-known Web API public repository.

1 Introduction

Many solutions have been recently proposed for supporting Web API search and mashup composition, as a new paradigm for agile web application development. When the aim of a web developer is to deploy a new application quickly and with a limited expense in terms of development time and costs, mashups represent a sustainable solution, fueled by on-line repositories, that offer large catalogues of Web APIs to be selected and composed. Nowadays, two main research fields have been carried out concerning mashups: (i) advanced solutions for mashup development, by providing technologies, models and CASE tools to ease the web application designers in aggregating the component Web APIs, setting the interactions between them and generating the glue code required to deploy the mashup application [13]; (ii) solutions to search for and rank relevant APIs out of a large number of available ones, registered within public repositories [7,10,14]. As the number of available Web APIs increases, alleviating the burden of finding the right ones is of paramount importance for mashup designers, and can be pursued through proper searching and ranking strategies. In [5] we proposed a framework for Web API search and ranking (WISER), that takes into account the descriptive features of the Web APIs (e.g., categories,

user-assigned semantic tags [4], technical features adopted by Web APIs, such as protocols and data formats) and their co-occurrence in existing mashups. In the first step, Web APIs are searched by specifying a category, a set of keywords or tags, optionally a set of technical features. In the next steps, choosing a new component means also taking into account other APIs already inserted in the mashup under construction, thus relying on past occurrences of the same APIs within existing applications to properly rank search candidates. The joint use of similarity techniques applied to descriptive features and information on Web API co-occurrence has been proved to be effective in [14]. In WISeR, we viewed Web API co-occurrence information as part of the designers' experience in using the available Web APIs in their own mashups. According to such an experience perspective, mashup designers learn from choices made by other designers in selecting available APIs for mashup development. This enables to obtain fruitful suggestions on which APIs could be more suitable for the mashup that is being developed. In this context, with designers we refer to mashup developers, who use and rate Web APIs to develop their own web applications, distinguishing them by Web API providers, that is, third parties who make Web APIs available on the network. The use of the wisdom of the other developers is getting more and more relevant for Web API discovery, in particular when API descriptions alone are not enough to identify the best API for a given set of requirements, because of the lack of precision or completeness in the descriptions themselves [14]. Nevertheless, questions remain unanswered on how designers' experience might be properly estimated and where it can be exploited within the Web API search and ranking process. In this paper, we introduce some innovations compared to our previous work on WISeR, that can be summarized as follows:

- we propose a set of metrics and techniques for the estimation of the designers' experience gained for mashup development; according to our knowledge, experience estimation is an innovative issue compared to existing approaches on Web API search and ranking;
- we describe how and where the designers' experience influences the WISeR search and ranking process;
- we present a preliminary evaluation of the proposed approach, based on the contents extracted from **ProgrammableWeb**, a well known public repository of Web APIs and mashups; experiments proved the feasibility, through the integration of designers' experience within WISeR, of getting finer comparisons between APIs, detecting differences in Web API rank otherwise undistinguished using traditional ranking capabilities.

The paper is organized as follows. In Section 2 we formulate the problem statement and we motivate our approach with an example. Section 3 describes the Web API model, on which our work is based. In Section 4 we discuss how to estimate the designers' experience and how to integrate it within the Web API search and ranking process. Implementation and evaluation issues are discussed in Section 5. In Section 6 the state of the art on Web API ranking for searching purposes is discussed and the cutting-edge features of our approach are highlighted. Finally, conclusions and future work in Section 7 close the paper.

2 Problem Statement and Motivations

To present the contributions described in this paper, we consider, as a running example, a temporary work agency aiming at designing a special session of its Web site to promote job offers, collect comments and job requests posted by people through their social network accounts and enable autonomous search for jobs, displaying on a map the job locations. For the running example, we rely on the contents of the ProgrammableWeb API repository as the most used and updated one. The temporary work agency needs Web APIs for posting and displaying comments from social networks (such as Twitter or LinkedIn), Web APIs for searching details on a specific job (such as Simply Hired Jobs), a map to display and mash up the different sources of information (e.g., GoogleMaps or BingMaps), to be integrated with agency APIs, that extract information about job offers from the agency information system.

The designer who is in charge of developing the new mashup on behalf of the temporary work agency may treasure the experience of other designers in using the same or similar Web APIs for mashups developed within very close situations, after checking that Web APIs fit search requirements such as categories, tags

The screenshot shows the ProgrammableWeb website interface. At the top, there is a navigation bar with links to API News, API Directory, For API Providers, For Developers, Listings, and Forum. Below the navigation bar, a banner features the text "APIs are the key to delivering on digital strategies" and a "Find out why" button. To the right of the banner are icons for Diversity, eBook, and eBOOK. The main content area has a heading "ProgrammableWeb: Showcasing Mashups and Apps by the APIs THEY CONSUME". It includes a search bar and a "Search Mashups" button. Below the search bar is a "Filter Mashups" section with a "By Category" dropdown menu containing "Google Maps" and "LinkedIn", and a checkbox for "Include Deprecated". A "Track this Search" button is located above a table of search results. The table columns are "Mashup Name", "Description", "Category", and "Date". The results listed are:

Mashup Name	Description	Category	Date
USF Map	Searchable Map Of The University Of South Florida (USF) Campus.	Mapping	06.14.2014
Reviewing.net	Reviewing.net Utilizes Rating APIs To Provide Users With A Grouping Of Individual Ratings From...	Search	06.11.2014
Arab World Weather	Arab World Weather Provides Up To Date Weather Forecasts Of Over All The Major Cities In The Arab...	Weather	05.22.2014
Goolzoom France	Goolzoom Uses Data From The Geopointtail And Google Maps API To Provide A Searchable Map. When...	Mapping	05.20.2014
Egypt Forecast	Egypt Forecast Provides Weather Conditions Of Over 70 Egyptian Cities Using Google Maps And...	Weather	05.12.2014

On the right side of the main content area, there is a sidebar titled "Search API Directory" which says "Search over 11,517 APIs updated daily". It includes a search bar, a "Browse by Category" button, a "Newest APIs" button, and a "Latest Mashups" button. Below the sidebar is a green button labeled "Add an API +". At the bottom right, there is a chart titled "Growth of Top 10 Web API Categories Since 2009" with data for 2013 and 2009.

Fig. 1. The ProgrammableWeb Web API Search interface

or technical features specified in the request. For example, if we consider the **ProgrammableWeb** repository, two Web APIs such as **Twitter** and **GoogleMaps** have been used together in about 500 mashups, while **LinkedIn** and **GoogleMaps** APIs have been used in no more than 30 mashups. This may suggest the use of **Twitter** API for posting and displaying comments from social networks, instead of **LinkedIn**, in a mashup where **GoogleMaps** has been already inserted, since the joint use of these APIs has been experienced more times and likely there are multiple examples on how to use these Web APIs together. In literature, metrics such as Web API popularity (computed as the number of mashups where the Web APIs have been used) and Web API co-occurrence have been already investigated to rank APIs for selection purposes [7,10,14], going beyond basic sorting facilities made available by **ProgrammableWeb** repository (see Figure 1). In WISER [5] we proposed an innovative Web API search framework, composed of:

- a Web API model**, organized according to three different perspectives, namely the *component* (Web APIs), the *application* (mashups built with Web APIs) and the *experience perspective* (designers who used Web APIs to develop their own mashups);
- a search model**, that has been designed to guide the Web API request formulation; specifically, the model is articulated as a *search target*, namely the development of a new mashup, the completion of an existing mashup or the substitution of a Web API in a given mashup, and a *search mode*, depending on the way the request is formulated (that is, which are the elements composing the request);
- a set of similarity metrics**, based on Web API model, to quantify the compliance of available Web APIs with respect to the specified request.

The combination of different types of similarity metrics, properly weighted for search and ranking, leveraging all the elements that describe a Web API (categories, semantic tags and votes assigned by mashup designers, who used the Web API, features of the mashups where the API has been used), and strategies to setup weights have been the innovative aspects of the WISER approach. Another distinctive feature in WISER is the inclusion within the Web API model of the skill in mashup development associated to each designer. The number of times a Web API has been used is doubtless relevant for Web API ranking, but the experience or skill of designers who used the Web API in their mashups should be properly taken into account. For instance, let's suppose that the average skill of developers who designed the mashups where **LinkedIn** and **GoogleMaps** APIs have been used together is significantly greater than the average skill of developers who designed mashups where **GoogleMaps** API has been used with **Twitter**. Therefore, the fact that **Twitter** and **GoogleMaps** APIs have been used together more frequently than **LinkedIn** and **GoogleMaps**, that would suggest the use of **Twitter** in a mashup under construction, where **GoogleMaps** is already present, should be properly balanced with the information about designers' experience. In WISER the skill is self-declared by the designer, according to a discrete scale:

1.0 for **expert**, 0.8 for **high confidence**, 0.5 for **medium confidence**, 0.3 for **low confidence** and 0.0 for **unexperienced**. However, the designers' experience should be estimated considering several aspects, such as the reputation and popularity of mashups developed by the designers themselves in the past. Hereafter, with *mashup reputation* we will refer to the rating of mashup based on votes assigned by other developers, while with *mashup popularity* we will refer to the success of mashup according to the Internet traffic around it (that, for instance, can be quantified through the Alexa.com traffic rank¹). Therefore, votes assigned by designers to mashups, exploited to determine their reputation, should be weighted with the designers' credibility, which may vary as influenced by multiple factors [12].

In this paper, we will address the following open issues:

- we will provide techniques and metrics to estimate the experience of designers in using Web APIs for mashup development; given a designer d and a set of mashups \mathcal{M} developed by d , we will estimate the experience of d in terms of the reputation and popularity of mashups in \mathcal{M} ; to the best of our knowledge, existing solutions relying on Web API popularity, co-occurrence and votes assigned to Web APIs and mashups do not provide any measure for estimating the designers' experience;
- given a mashup m , with an associated reputation, inferred in terms of votes assigned to the mashup, we will present techniques and metrics to extend the reputation assessment, considering, among the other aspects, the credibility of designers in assigning their votes;
- we will discuss how to integrate the new metrics within the WISER similarity measures for Web API search and ranking.

3 Web API Model

In this paper, we refer to the Web API model proposed in [5]. Web APIs are modeled as a set of properties, partially extracted from the **ProgrammableWeb** repository. Among these properties, the categories belong to a pre-defined, top-down classification within the repository, while tags are assigned in a bottom-up fashion by the designers who use the repository to select and rate APIs. For example, in Table 1(a) a partial description of the **LinkedIn** API is shown. We refer to protocols, data formats, SSL support, authentication mechanism shown in Table 1(a) as *technical features*.

Specifically, most of the properties listed in Table 1(a) are extracted from the **ProgrammableWeb** repository, except for semantic tags for which a proper interface has been implemented in the WISER system on top of the repository, to support the designers in semantic tagging. Semantic tags are adopted to face synonymy and homonymy of traditional tags. During the assignment of such tags, sense disambiguation techniques based on the WordNet lexical system are applied, as extensively described in [4]. In WordNet the meaning of terms is

¹ <http://alexa.com>.

Table 1. Partial description of the LinkedIn API used for the running example (a) and of a mashup where the LinkedIn API has been used (b), as extracted from the ProgrammableWeb repository

Web API name	LinkedIn
Web API URI	https://developer.linkedin.com/
Categories	Social
Designers-assigned semantic tags	{social, {sociable, mixer} “a party of people assembled to promote sociability and communal activity”}
Protocols	{REST, Javascript}
Data formats	{XML, JSON, JSONP}
SSL support	{yes}
Authentication mechanism	{OAuth}

(a)

Mashup name	Relocator
Mashup URI	http://6flex.com/relocator/
Mashup composition (Web APIs)	{Facebook, GoogleMaps, Indeed, LinkedIn}
Designers-assigned semantic tags	{employment, {employ} “the state of being employed or having a job”} {job, {occupation, business, line of work, line} “the principal activity in your life that you do to earn money”}

(b)

defined by means of *synsets*. Each synset has a human readable definition and a set of synonyms. Starting from the tag specified by the designer, WordNet is queried, all the synsets that contain that term are retrieved and the designer is asked to select the intended meaning. In our model, each semantic tag is a triplet, composed of the term itself extracted from WordNet, the set of all the terms in the same synset and the human readable definition associated with the synset (see, for example, the tag **social** in Table 1(a)).

The ProgrammableWeb repository also enables to retrieve information about the set of mashups where a Web API has been included. An example of mashup description is shown in Table 1(b). The set of semantic tags that describe a mashup are defined in the same way as Web API semantic tags. No technical features are provided for mashups, since these features are directly related to the component Web APIs.

Our Web API model also includes the designers who used the Web APIs to build their own mashups. Designers may assign votes to the Web APIs. In a context, such as the one of mashup developers, where designers exchange their experiences in using Web APIs, votes become an enabling feature for realizing this kind of collaboration. In this sense, all the most popular Web API repositories include a rating system. In WISER, votes are assigned by the designer with the support of the NHLBI 9-point Scoring System, whose adoption for Web API rating has been described in [4] and is reported in Table 2 for convenience. The designer is guided by qualitative ratings, that classify Web APIs to be scored. The WISER system introduces an important distinction for Web API rating,

taking into account the context in which Web APIs have been used, that is, the mashups that include the Web APIs. Votes assigned to Web APIs are contextualized with respect to the different mashups where Web APIs have been rated. For instance, the **LinkedIn** API might be more relevant for a professional application in job search compared to the **Twitter** API. Nevertheless, since a high score not necessarily implies good quality of the resources that have been rated [12], in our approach we will integrate our rating system with similarity and co-occurrence metrics and we will propose a way to estimate the developers' reputation in the context of Web API selection.

Table 2. The 9-point Scoring System for the classification of designers' Web API rating (from [4])

Qualitative rating (additional guidance on strengths/weaknesses)	Votes
POOR (<i>completely useless and wrong</i>)	0.2
MARGINAL (<i>several problems during execution</i>)	0.3
FAIR (<i>slow and cumbersome</i>)	0.4
SATISFACTORY (<i>small performance penalty</i>)	0.5
GOOD (<i>minimum application requirements are satisfied</i>)	0.6
VERY GOOD (<i>good performance and minimum application requirements are satisfied</i>)	0.7
EXCELLENT (<i>discreet performance and satisfying functionalities</i>)	0.8
OUTSTANDING (<i>very good performances and functionalities</i>)	0.9
EXCEPTIONAL (<i>very good performances and functionalities and easy to use</i>)	1.0

3.1 Request Formulation

The Web API model summarized in this section is used to guide the designer in the formulation of the Web API request \mathcal{W}^r as well, formally represented as $\mathcal{W}^r = \langle c_{\mathcal{W}}^r, \{t_{\mathcal{W}}^r\}, T_{\mathcal{W}}^r, [M^r] \rangle$, where: (i) $c_{\mathcal{W}}^r$ is the required category; (ii) $\{t_{\mathcal{W}}^r\}$ is the set of required tags, semantically characterized through the word sense disambiguation technique presented in [4]; (iii) $T_{\mathcal{W}}^r$ are the required technical features, to be distinguished among protocols, data formats, SSL support and authentication mechanism; for each kind of feature, among the ones shown in Table 1(a), a distinct vector is inserted in the request \mathcal{W}^r (see the example below); (iv) M^r is the mashup where the Web API to search for will be inserted, in turn characterized through a set of semantic tags $\{t_M^r\}$ and the set of Web APIs $\{\mathcal{W}_M^r\}$ already included in the mashup M^r . The presence of the M^r specification in the request \mathcal{W}^r depends on the search target. If the target is to select a single Web API (for instance, to start a new mashup), then no APIs $\{\mathcal{W}_M^r\}$ have been previously selected yet and M^r is not specified. Otherwise, if the target is the completion of an existing mashup or the substitution of a Web API in a given mashup, then M^r represents the mashup where the Web API that is being searched will be inserted. In [5] a more in depth analysis of the relationship between the search target and \mathcal{W}^r formulation is presented. The few hints given here are enough to understand the rest of the present discussion.

Example. A request \mathcal{W}^r for finding a Web API for job search, to be used in a mashup designed for recruitment, which already includes the GoogleMaps and LinkedIn APIs, might be represented as follows, where it is specified that a Web API that adopts the REST protocol and XML and JSON data formats is required:

$$\begin{aligned}\mathcal{W}^r &= \langle c_{\mathcal{W}}^r = \text{Job Search}; \\ \{t_{\mathcal{W}}^r\} &= \{\langle \text{job}, \{\text{occupation, business, line of work, line}\}, \text{"the principal activity in your life that you do to earn money"} \rangle\}; \\ T_{\mathcal{W}}^r.\text{protocols} &= \{\text{REST}\}; T_{\mathcal{W}}^r.\text{dataFormats} = \{\text{XML, JSON}\}; \\ M^r &= \langle \{t_M^r\} = \{\langle \text{recruitment, \{enlisting\}}, \text{"(the act of getting recruits; enlisting people for the army (or for a job or a cause etc.)"} \rangle\}; \{\mathcal{W}_M^r\} = \{\text{GoogleMaps, LinkedIn}\} \rangle\end{aligned}$$

4 Integrating Designers' Experience in Web API Selection

4.1 Web API Similarity Metrics

A set of similarity metrics have been designed to compare the elements of Web API descriptions $\{\mathcal{W}\}$ extracted from the ProgrammableWeb repository and properly tagged, against the corresponding elements of the request \mathcal{W}^r . The rationale behind similarity metrics is that the more a Web API description \mathcal{W} from the repository fits the request \mathcal{W}^r , the more their categories, their semantic tags, their technical features and the mashups that contain them are similar. The building blocks are the *category similarity*, the *semantic tag similarity*, the *technical feature similarity* and the *mashup composition similarity* metrics:

- the **category similarity** between the category $c_{\mathcal{W}}^r$ of \mathcal{W}^r and the category $c_{\mathcal{W}}$ of \mathcal{W} , denoted with $Sim_{cat}(c_{\mathcal{W}}^r, c_{\mathcal{W}}) \in [0, 1]$, measures the degree of overlapping between the two categories, in terms of the number of Web APIs that are categorized in both the categories compared with the overall number of Web APIs classified in $c_{\mathcal{W}}^r$ and in $c_{\mathcal{W}}$;
- the **semantic tag similarity** between two sets of semantic tags, denoted with $Sim_{tag}() \in [0, 1]$, is computed by evaluating the term affinity, based on WordNet, between pairs of tags, one from the first set and one from the second set, and by combining them through the Dice formula, as extensively described in [4]; semantic tag similarity is used both to compare semantic tags associated to Web APIs and to compare semantic tags associated to mashups;
- the **technical feature similarity** between two sets of homogeneous features, one from the request \mathcal{W}^r and one from the Web API description \mathcal{W} , is computed as the number of common feature values compared with the overall number of feature values in \mathcal{W}^r and \mathcal{W} ; a distinct type of similarity is designed for each kind of feature, such as protocols, data formats and authentication mechanisms; comparison based on the SSL support is even simpler, since it must be based on a true/false value only; for example, the similarity of \mathcal{W}^r and the LinkedIn API description shown in Table 1(a) with

respect to the data formats is computed as $\frac{2 \cdot |\{\text{XML, JSON, JSONP}\} \cap \{\text{XML, JSON}\}|}{|\{\text{XML, JSON, JSONP}\}| + |\{\text{XML, JSON}\}|} = 0.8$; in [5] an in-depth discussion on variants of this measure to be applied for different search targets is provided; such a discussion is out of the scope of this paper;

- the **mashup composition similarity** between the mashup M^r in the request, composed of a set $\{\mathcal{W}_M^r\}$ of Web APIs, and another mashup composed of a set $\{\mathcal{W}_k\}$ of Web APIs, denoted with $Sim_{comp}(\{\mathcal{W}_M^r\}, \{\mathcal{W}_k\})$, measures the degree of overlapping between the two compositions, evaluated as the number of common Web APIs in the two mashups, following the same rationale of $Sim_{cat}()$ computation.

The overall similarity measure between the request \mathcal{W}^r and each API \mathcal{W} , denoted with $Sim(\mathcal{W}^r, \mathcal{W}) \in [0, 1]$, is computed as:

$$Sim(\mathcal{W}^r, \mathcal{W}) = \omega \cdot APISim(\mathcal{W}^r, \mathcal{W}) + (1 - \omega) \cdot \left(\frac{\sum_m \sigma_m \cdot MashupSim(M^r, M_m)}{|\mathcal{M}|} \right) \quad (1)$$

$APISim(\mathcal{W}^r, \mathcal{W}) \in [0, 1]$ denotes the similarity between the request \mathcal{W}^r and the Web API description \mathcal{W} as a linear combination of category similarity, technical feature similarity and the similarity between the semantic tags specified in \mathcal{W}^r and the semantic tags assigned to \mathcal{W} . The setup of weights used for such a linear combination depends on the search model and is detailed in [5]. $MashupSim(M^r, M_m) \in [0, 1]$ denotes the similarity between the mashup $M_m \in \mathcal{M}$, among mashups \mathcal{M} where \mathcal{W} has been used, and the mashup M^r under construction, where the Web API that is being searched will be included. $MashupSim()$ value is computed as a linear combination of mashup composition similarity between M_m and M^r and the similarity between the semantic tags specified for M^r and semantic tags assigned to M_m . The setup of weights used for such a linear combination depends on the search model and is detailed in [5]. The weight ω is set to balance the Web API similarity and mashup similarity in $Sim()$ computation. Since we consider the two terms as equally relevant, we set $\omega = 0.5$, but it can be modified by the designer who is searching for the Web API. In particular, the second term of the linear combination in Equation (1) takes into account that the Web API \mathcal{W} has been used in different mashups, that may have different reputation (according to the votes assigned to them by other designers) and popularity (according to the internet traffic on the mashup URL), as we will discuss in the following.

4.2 Estimating the Mashup Reputation and Popularity

Mashup reputation and popularity are estimated through the weight $\sigma_m \in [0, 1]$. Specifically, $\sigma_m = 1$ means that m -th mashup $M_m \in \mathcal{M}$ has the maximum reputation and popularity. The second term of the linear combination in Equation (1) ensures that best rated and most popular mashups have a higher impact on the $Sim()$ computation. Intuitively, the closer the σ_m and $MashupSim()$ values to 1 (maximum value) for all the mashups in \mathcal{M} , the closer the second term in

Equation (1) to 1.0. The use of σ_m for weighting the mashup similarity in $Sim()$ computation is one of the contributions of this paper compared to our previous work [5], where mashup similarity was weighted using the development skill of the designer who owns the mashup. The skill was self-declared by the designer on a discrete scale: 1.0 for **expert**, 0.8 for **high confidence**, 0.5 for **medium confidence**, 0.3 for **low confidence** and 0.0 for **unexperienced**. In this paper, we propose the application of techniques to estimate σ_m as follows:

$$\sigma_m = \omega_u \cdot reputation(M_m) + \omega_t \cdot trafficRank(M_m) \in [0, 1] \quad (2)$$

$reputation(M_m)$ denotes the reputation of mashup M_m , $trafficRank(M_m)$ is used to estimate the popularity of M_m based on the internet traffic, the weights $\omega_u, \omega_t \in [0, 1]$, where $\omega_u + \omega_t = 1.0$, are used to balance the two terms used to estimate σ_m . The votes are assigned by mashup designers, who have a skill for the application domain that is higher than the average skill of network users who visit the mashup URL, as detected by the Alexa traffic rank measure. Therefore, we have chosen $0 \leq \omega_t < \omega_u \leq 1.0$. Specifically, we set $\omega_t \approx 0.4$ and $\omega_u \approx 0.6$. This choice has been validated through a set of preliminary experiments as shown in Section 5.

The $reputation(M_m)$ is computed according to the votes assigned to the mashup by other designers on WISer:

$$reputation(M_m) = \frac{1}{k} \cdot \sum_{i=1}^k vote(d_i, M_m) \cdot credibility(d_i) \in [0, 1] \quad (3)$$

where $vote(d_i, M_m) \in [0, 1]$ is the vote assigned by i-th designer d_i to the mashup M_m , k is the total number of votes assigned to M_m . In Equation (3) votes are properly weighted with the designer's credibility $credibility(d_i) \in [0, 1]$, where 1.0 means the maximum credibility. Weighting the credibility of an user on the Web, even in a specific application domain such as the one we are considering in this work, is not straightforward. Several reputations models have been provided in the past. In this paper, we refer to the approach described in [12], that has been focused on providing a solution for assessing the reputation of service providers under conditions where the resources to rate are autonomously provided, highly volatile and a-priori unknown. Properly extended, the approach can be used in different domains as well. In Section 4.4 we will show how we adapted to the problem of Web API selection some of the concepts expressed in [12].

The $trafficRank(M_m)$ is computed according to the Internet traffic concerning the mashup, based on the concept that the higher the traffic rank of a Web application, the higher its popularity. To this aim, we rely on the Alexa traffic rank $rank(M_m)$, where the highest rank is equal to 1. Since the Alexa rank is calculated on the Web in general, to compute the rank of a mashup $M_m \in \mathcal{M}$ compared to the rank of the other mashups in \mathcal{M} , we normalized the rank with respect to the highest one in \mathcal{M} , obtaining the value as computed in the following:

$$\text{trafficRank}(M_m) = \frac{\max_{j=1}^{|\mathcal{M}|} \{ \text{rank}(M_j), M_j \in \mathcal{M} \}}{\text{rank}(M_m)} \in [0, 1] \quad (4)$$

For example, consider the values of Alexa $\text{rank}()$ for the mashups where the **Simply Hired Jobs** API has been included. Such values are shown in Table 3. For instance, $\text{trafficRank}(\text{InstantJobSearch}) = 7,558,062 / 11,946,551 = 0.63$. The other values for $\text{trafficRank}(M_m)$ are shown in the third column of the table.

Table 3. The values of Alexa rank for the mashups where the **Simply Hired Jobs** API has been included

M_m	$\text{rank}(M_m)$	$\text{trafficRank}(M_m)$
Instant Job Search	11,946,551	0.63
AllJobs.biz	7,558,062	1
Classified on Google Maps and Charts	11,625,758	0.65
Trackle	18,275,821	0.41

4.3 Web API Ranking

The Web APIs included in the search results (that we denote with $\{\mathcal{W}'\} \subseteq \{\mathcal{W}\}$) are those whose overall similarity $\text{Sim}(\mathcal{W}', \mathcal{W}) \geq th$, where th is set by the designer who is looking for Web APIs. Each Web API in $\{\mathcal{W}'\}$ is ranked taking into account both its overall similarity compared to the request \mathcal{W}^r and the reputation of the Web API based on votes assigned by the other designers. In particular, the two elements are combined in the following harmonic mean:

$$\rho(\mathcal{W}') = \frac{2 \cdot \text{reputation}(\mathcal{W}') \cdot \text{Sim}(\mathcal{W}^r, \mathcal{W}')}{\text{reputation}(\mathcal{W}') + \text{Sim}(\mathcal{W}^r, \mathcal{W}')} \in [0, 1] \quad (5)$$

where $\rho : \{\mathcal{W}'\} \mapsto [0, 1]$ is the *ranking function* and $\text{reputation}(\mathcal{W}')$ is computed following the same rationale of the mashup reputation (Equation (3)), that is, it is weighted with the credibility of designers who rated Web APIs:

$$\text{reputation}(\mathcal{W}') = \frac{1}{n} \cdot \sum_{i=1}^n \text{vote}(d_i, \mathcal{W}') \cdot \text{credibility}(d_i) \in [0, 1] \quad (6)$$

where $\text{vote}(d_i, \mathcal{W}') \in [0, 1]$ is the vote assigned by i-th designer d_i to the Web API \mathcal{W}' , n is the total number of votes assigned to \mathcal{W}' . We remark here that also votes used for Web API reputation are those assigned by mashup designers, who adopted the APIs to develop their own applications or who used the mashups where APIs have been included.

4.4 Extending the Reputation Assessment

The measures exposed above may be further extended considering the conditions under which the reputation of Web APIs and mashups may vary. Specifically, such variability may depend on:

1. the variation of credibility of designers, derived from the deviation of votes assigned by the designer d_i from the majority opinion extracted from all the votes assigned to a Web API or a mashup, and from the coherency of votes assigned by the designer based on past ratings history, as specified in [12];
2. the feedbacks of designers who used the votes to search for a Web API; such feedbacks can be used as well to adjust the credibility of a designer according to the deviation between the votes and the personal experience of other designers who used the votes themselves for search purposes;
3. the temporal sensitivity of designers' votes, where older votes are given less weight than more recent ones;
4. the contextualization of votes assigned to a Web API with respect to a specific mashup.

Designers' Credibility. In our approach, we use the deviation of votes of the designer d_i from the majority opinion as proposed in [12]. The basic idea is that, if the reported vote does not agree with the majority opinion, the designer's credibility is decreased, otherwise it is increased. The k-mean clustering algorithm is used on the set of votes assigned by the designer d_i to a Web API or a mashup. The majority opinion is represented by the most densely populated cluster, whose centroid is considered as the majority rating:

$$M = \text{centroid}(\max_{i=1}^k(C_i)) \quad (7)$$

where k is the total number of clusters, $\max()$ returns the cluster with the largest membership and $\text{centroid}()$ computes the centroid of the cluster. The Euclidean distance between a $\text{vote}(d_i, X)$ assigned by the designer d_i to a Web API or mashup X and the centroid M , denoted with $\text{dist}(\text{vote}(d_i, X), M)$, is used to adjust the d_i credibility each time the designer d_i assigns a vote. We remark that in our approach the designer d_i may assign votes to either a Web API ($X \equiv \mathcal{W}'$, see Equation (6)) or a mashup ($X \equiv M_m$, see Equation (3)): in the two cases, two different $\text{dist}()$ values are computed.

The coherency of votes assigned by the designer d_i based on past rating history can be inferred if the designer assigns the same or very close votes to the same Web API or the same mashup when the conditions under which vote is assigned do not change. This means that two conditions must hold: (1) a designer should be able to assign more than one vote to a single Web API or mashup; (2) the system should enable to quantify the circumstances under which votes are assigned. In our approach, these conditions do not hold. Therefore, a measure of coherency has not been considered at this development stage. Similarly, also the feedbacks of designers who used the votes assigned by d_i to search for a Web API are not allowed in the **ProgrammableWeb** repository and therefore have not been implemented in WISeR yet for compatibility purposes. However, these aspects will be further investigated as future work. Peculiar aspects of our approach are the following ones.

Temporal Sensitivity of Votes. In order to take into account the obsolescence of the votes assigned by a developer d_i to a Web API or a mashup, we extend our

approach considering the age of each $vote(d_i, X)$, denoted with $age(vote(d_i, X))$, computed since the timestamp when the vote has been assigned. The obsolescence factor associated to $vote(d_i, X)$ is computed as follows:

$$obs(vote(d_i, X)) = \frac{1}{2^{age(vote(d_i, X))}} \in [0, 1] \quad (8)$$

In this way, for the most recent votes, $obs(vote(d_i, X)) \approx 1.0$, while it decreases exponentially till $obs(vote(d_i, X)) \approx 0.0$.

Contextualization of Web API Votes. The WISER system enables designers to contextualize their votes assigned to a Web API in a specific mashup. In this case, the reputation formula for a Web API \mathcal{W}' must be changed as follows:

$$reputation^{ext}(\mathcal{W}') = \frac{1}{n} \cdot \sum_{i=1}^n \left(\frac{\sum_m vote(d_i, \mathcal{W}', M_m) \cdot credibility(d_i) \cdot MashupSim(M^r, M_m)}{|\{M_m\}|} \right) \quad (9)$$

where $reputation^{ext}(\mathcal{W}') \in [0, 1]$, $vote(d_i, \mathcal{W}', M_m)$ represents the vote assigned by designer d_i to the Web API \mathcal{W}' with respect to the mashup M_m . The product on numerator in Equation (9) is motivated by the fact that a designer d_i may assign several votes to the Web API in different mashups. The rationale behind the equation is that, the closer $MashupSim(M^r, M_m)$ to 1.0, the more relevance is given to the vote assigned by the designer to the Web API.

5 Implementation and Experimental Evaluation

Figure 2 shows the functional architecture of the WISER system, where the ranking facilities described in this paper, enhanced with the estimation of designers' experience, have been integrated. To access some functionalities of the system, namely advanced Web API search, Web API rating and semantic tagging, designers register themselves in the system and may log in using their own credentials.

The system is based on the *Enriched Web API Registry*, where Web APIs, mashups descriptions, estimated designers' experience and votes are stored. However, in the registry, not all the details about Web APIs and mashups are stored, since they may be partially retrieved from the *ProgrammableWeb* repository through proper methods², invoked within the *ProgrammableWeb Data Access Module*. This makes the WISER system compatible with the *ProgrammableWeb* repository.

The main features of WISER have been implemented as Web services. This ensures high modularity and platform-independency. For instance, the *Word Sense disambiguation Module* has been implemented in Java to exploit the most reliable version of WordNet, that has been made available for the J2EE platform, and at the same time to ease the communication with the WISER front-end, that has

² <http://api.programmableweb.com/>

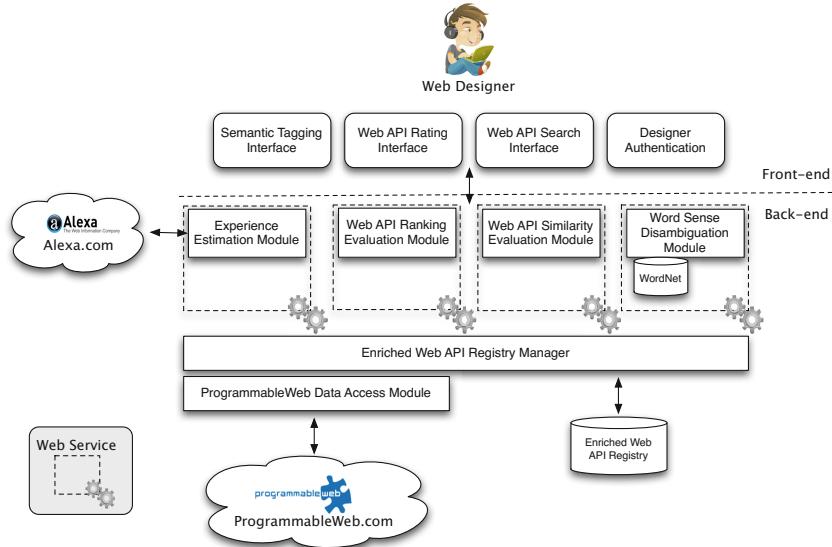


Fig. 2. The WISeR functional architecture

been implemented as a set of PHP pages. Similarly, the other Web services have been deployed on the Java platform. The metrics for the estimation of designers' experience have been implemented within the *Experience Estimation Module*, that interacts with methods of the **Alexa.com** ranking service. The front-end of the system is in charge of enabling designers to register and authenticate themselves (*Designer Authentication*), to perform keyword-based Web API search by relying on **ProgrammableWeb** methods, to perform advanced Web API search and ranking, being properly guided by a wizard in the formulation of the request (*Web API Search Interface*), to rate Web APIs according to the 9-point scoring system, assigning votes both with and without contextualization with respect to the mashups (*Web API Rating Interface*), to perform semantic tagging (*Semantic Tagging Interface*).

We run a set of preliminary experiments to test the effectiveness of engaging the designers' experience estimation in Web API ranking. These experiments prove the feasibility, through the integration of designers' experience within WISeR, of getting finer comparisons between APIs, detecting differences in Web API rank otherwise undistinguished using traditional ranking capabilities. Experiments are based on the contents of the **ProgrammableWeb** repository. In particular, they focus on the application domain of the running example. We manually looked for mashups and Web APIs on job search from the repository, using the keywords "job", "job search", their WordNet synonyms (e.g., "occupation", "recruitment", "business", "work") and the category "Job Search". We obtained a set of 65 Web APIs, included in 61 mashups.

We then randomly chose 10 mashups among search results. For each mashup $M_i, i = 1..10$, we extracted a Web API $\mathcal{W}_i, i = 1..10$. We then issued a request using the features of \mathcal{W}_i given a mashup $M'_i = M_i / \{\mathcal{W}_i\}$ and we compared the ranking values for the obtained Web APIs with respect to: (a) the Web API votes, as assigned within the **ProgrammableWeb** repository; (b) basic WISer ranking capabilities, according to Equation (6); (c) contextualized WISer ranking capabilities, according to Equation (9).

Figure 3 shows $\rho(\mathcal{W}')$ values for the sample request \mathcal{W}'^r used in the running example (Section 2). We note that **ProgrammableWeb** does not allow to distinguish many Web APIs with respect to the request \mathcal{W}'^r , while WISer ranking capabilities enable to highlight significative differences. For example, rank of **Zipster**, **USASearchJobs** and **SimplyHiredJobs** seem comparable in **ProgrammableWeb**, while WISer reveals that the latter should be ranked better. By inspecting the Web APIs and mashups where they have been included, we verified that the similarity of mashups where **SimplyHiredJobs** has been used, compared to M'^r , is higher than analogous similarities for **USASearchJobs** and **Zipster**, respectively.

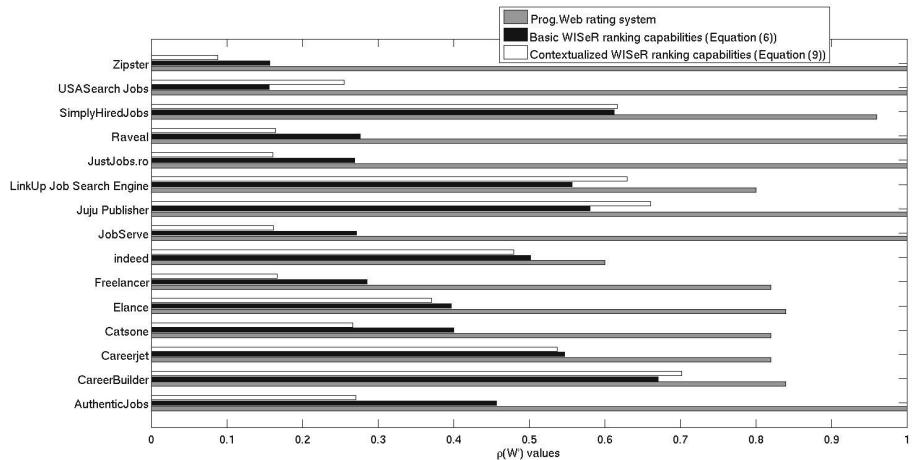


Fig. 3. Comparison between the rank results obtained using **ProgrammableWeb** and the WISer ranking system

Experimental results shown in Figure 3 have been obtained after performing a setup of ω_u and ω_t weights, used to properly balance the votes assigned by designers and the Alexa traffic rank measure. The setup has been performed as follows. We considered a pair of mashups M_i and M_j and their votes assigned by a team of expert designers, who agreed upon their votes. We then computed the Alexa traffic rank and we varied the values of ω_u and ω_t in the $[0, 1]$ range until the value of $reputation(M_i)$ compared to the value of $reputation(M_j)$ reflected the order imposed to M_i and M_j by the votes assigned by the team of expert

designers. We repeated the experiment for other pairs of mashups and we considered the average results, obtaining $\omega_t \approx 0.4$ and $\omega_u \approx 0.6$. However, during Web API search in WISeR, these weights can be modified by the designer as well.

6 Related Work

In literature, research on resource discovery over the web clearly distinguishes between web service discovery, that, given its machine-to-machine nature, is addressed as automatic selection and composition of software components [2], and Web API selection, that is more user-oriented and aims at supporting developers to select the right APIs to build their own mashups [1]. In this sense, features like WSDLs are not considered as relevant for Web APIs as for Web services and are not made available in Web API repositories. This explains why similarity metrics for Web APIs, proposed in [5] and integrated here with additional information to estimate designers' experience, exploit elements in Web API descriptions such as categories and tags only, that are more user-oriented. We proposed similarity metrics for web service discovery in our previous works [3,6]. A comparison with other approaches with respect to similarity metrics was discussed in [5]. We performed here a comparison among approaches from the literature based on different aspects that have been jointly considered for Web API selection and are summarized in Table 4.

Table 4. Comparison between approaches for Web API ranking according to different kinds of measures

Measure → Approach ↓	keywords/categories	#times an API is used	technical features	votes by designers	Web API quality	Web API co-occurrence
ApiHut [10]	×	×	×	×		
PEUDOM [7]			×		×	×
MashupReco [14]	×	×				×
MatchUp [11]						×
MashupAdvisor [9]						×
Chowdhury et al. [8]						×
WISeR [5]	×	×	×	×		×

In [10], Web APIs are searched according to four different kinds of facets, namely functionality (based on categories and tags assigned by designers to Web APIs), messaging formats, programming language bindings and protocols. Moreover, a metric (called *Serviut rank*), based on the number of mashups that include the APIs, the network traffic related to such mashups and the votes assigned by users to the mashups within **ProgrammableWeb** are used to rank candidate APIs. PEUDOM [7] defines a set of quality parameters both at the component (e.g., languages and data formats enhancing operability and interoperability) and at the composite application level, where aggregated quality is achieved by combining the quality parameters of each single component. In [7] the so-called community-perceived quality is also proposed, where the most frequent associations among (categories of) components can be mined and exploited, since they

may reflect the designers' perception of the quality of the created mashups. The work in [7] does not distinguish among different mashups, to properly weight the skill of mashup owners. In MashupReco [14], Web APIs are organized according to their functionalities, extracted from their natural language description, and a network, where links are based on the co-occurrence of Web APIs in the same mashups. Also in this work the designers' skill is not estimated. In [11] a formal model based on Datalog rules is proposed to capture all the aspects of a mashup component (called *mashlet*). In this model, authors consider the mashups that have been implemented using the Web API that is being modeled: when the designer selects a mashlet, the system suggests other mashlets to be connected on the basis of recurrent patterns of components in the existing mashups. No votes assigned to Web APIs or mashups and designers' expertise are considered. The MashupAdvisor approach [9] recommends a set of possible outputs for a specific mashup: each output corresponds to some transformations of the data being manipulated by the mashup; therefore, statistical (co-)occurrence of input and output concepts are derived from existing compositions. Also in this case, designers' votes, possibly weighted through mashup development experience, are not considered. In the approach presented in [8], composition knowledge (i.e., knowledge that can be harvested from existing, successful mashups or compositions defined by other and possibly more skilled developers) is provided as advices associated with patterns of Web APIs; triggers state the conditions under which the advices are offered. The proposal relies on implicit semantics gathered by different mining techniques and statistical data analysis on existing compositions. An in-depth estimation of designers' skill is not performed. Exploitation of designers' expertise to weight their votes has been introduced in [5]. Compared to them, we introduced here the experience estimation as described in the previous sections.

7 Conclusions

In this paper, we proposed a set of metrics and techniques for the estimation of designers' experience in using Web APIs for mashup development, performed considering several aspects, such as the reputation and popularity of mashups developed by the designers themselves. Experience estimation has been exploited for supporting Web API search and ranking and some preliminary experiments, based on the contents of a well-known Web API public repository, have been also discussed. A more extensive experimental evaluation, involving a large set of mashup designers, is being performed. Future work will be also devoted to the study of new features to be implemented on top of the contents available within the public repository, namely the introduction of feedbacks of designers to properly evaluate the usefulness of other designers' suggestions and a refinement of Web API and mashup rating, where the designers would be able to explicitly specify different aspects they considered to rate Web APIs and mashups. This would enable the identification of personal preferences, thus implementing finer recommendation strategies.

References

1. Beemer, B., Gregg, D.: Mashups: A Literature Review and Classification Framework. *Future Internet* 1, 59–87 (2009)
2. Bianchini, D., De Antonellis, V., Melchiori, M.: Capability matching and similarity reasoning in service discovery. In: CEUR Workshop Proceedings, vol. 160 (2005)
3. Bianchini, D., De Antonellis, V., Melchiori, M.: Service-based semantic search in P2P systems. In: Proc. of 7th IEEE European Conference on Web Services (ECOWS), pp. 7–16 (2009)
4. Bianchini, D., De Antonellis, V., Melchiori, M.: Semantic Collaborative Tagging for Web APIs Sharing and Reuse. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 76–90. Springer, Heidelberg (2012)
5. Bianchini, D., De Antonellis, V., Melchiori, M.: A Multi-perspective Framework for Web API Search in Enterprise Mashup Design. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 353–368. Springer, Heidelberg (2013)
6. Bianchini, D., De Antonellis, V., Melchiori, M., Salvi, D.: Semantic-enriched service discovery. In: Proc. of the 22nd International Conference on Data Engineering (ICDE), pp. 38–47 (2006)
7. Cappiello, C., Matera, M., Picozzi, M., Daniel, F., Fernandez, A.: Quality-Aware Mashup Composition: Issues, Techniques and Tools. In: Proc. of 8th Int. Conference on Quality of Information and Communications Technologies (QUATIC 2012), pp. 10–19 (2012)
8. Chowdhury, S., Chudnovskyy, O., Niederhausen, M., Pietschmann, S., Sharples, P., Gaedke, F.D.M.: Complementary assistance mechanisms for end user mashup composition. In: Proc. of the 22nd International Conference on World Wide Web Companion, pp. 269–272 (2013)
9. Elmeleegy, H., Ivan, A., Akkiraju, R., Goodwin, R.: MashupAdvisor: A Recommendation Tool for Mashup Development. In: Proc. of 6th Int. Conference on Web Services (ICWS 2008), Beijin, China, pp. 337–344 (2008)
10. Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A., Verma, K.: A Faceted Classification Based Approach to Search and Rank Web APIs. In: Proc. of International Conference on Web Services (ICWS), pp. 177–184 (2008)
11. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: Proc. of the 35th Int. Conference on Very Large DataBases (VLDB), Lyon, France, pp. 538–549 (2009)
12. Malik, Z., Bouguettaya, A.: RATEWeb: Reputation Assessment for Trust Establishment among Web Services. VLBD Journal 18, 885–911 (2009)
13. Matera, M., Picozzi, M., Pini, M., Tonazzo, M.: PEUDOM: A mashup platform for the end user development of common information spaces. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 494–497. Springer, Heidelberg (2013)
14. Tapia, B., Torres, R., Astudillo, H.: Simplifying mashup component selection with a combined similarity- and social-based technique. In: Proceedings of the 5th International Workshop on Web APIs and Service Mashups, pp. 1–8 (2011)

Software Support Requirements for Awareness in Collaborative Modeling

Michel Dirix^{1,2}, Xavier Le Pallec¹, and Alexis Muller²

¹ University of Lille1 - France

michel.dirix@ed-univ-lille1.fr,

xavier.le-pallec@univ-lille1.fr

² Axellience - France

{michel.dirix,alexis.muller}@genmymodel.com

Abstract. To address issues of traditional modeling tools (installation, model versioning and lack of model repositories), Axellience has developed the first online UML modeling tool. In GenMyModel's beta-phase, the most requested feature was collaboration. Supporting collaborative modeling involves addressing classical concerns of CSCW. These issues are usually classified through core dimensions like awareness and articulation work. We decided to focus our research on the most important dimension: awareness. Commercial modeling tools and research prototypes provide little support for awareness. To define the importance of awareness in modeling tools, we decided to study what awareness information is really required in collaborative modeling and to assess its importance according to articulation work types. To do this, we have implemented a basic collaboration system without constraint on articulation work. After a few months of use, we have identified three articulation work types present in more than 500 collaborative projects. This preliminary study allowed us to define awareness elements potentially needed for each articulation work type. As these elements are different for each articulation work type, we launched different surveys for each one of them. With these surveys, we have sorted awareness information by relevance according to articulation work types.

Keywords: Modeling, Collaboration, Awareness, Survey.

1 Introduction

When a person uses a modeling tool, one problem is remembering where models are saved. Furthermore, installing a modeling tool can be time-consuming. When collaborators want to send models between them, they have to ensure that collaborators use the same version of the modeling tool. GenMyModel is an answer to these problems. GenMyModel¹ is an online modeling tool. As it is online, users do not have to install or update the modeling tool. In addition, they do not have to know where their models are saved. While GenMyModel

¹ <http://www.genmymodel.com/>

was in beta-phase and had only 200 users registered, collaboration was the most requested feature. It was not planned at the beginning of GenMyModel's development to provide collaboration so quickly. We started by adding a basic version of collaboration. As of mid-May, GenMyModel had more than 50,000 users distributed in 200 countries, and more than 5,800 users were involved in more than 3,400 collaboratives projects. Collaboration is widely used and confirms the need of real-time collaboration as requested when GenMyModel had 200 users. Collaboration is above all being aware of others' activities. It implies displaying information about these activities. Thus, supporting collaborative modeling involves addressing classical concerns of CSCW(Computer-Supported Cooperative Work). "CSCW should be conceived as an endeavour to understand the nature and characteristics of cooperative work with the objective of designing adequate computer-based technologies" [1]. CSCW contains a number of core dimensions like:

Articulation Work: It consists of articulating distributed work of users.[2].

Awareness: Awareness is one crucial aspect of switching between work in solitude and collaboration.[3]. Awareness provides important information about activities of collaborators and interactions with the shared workspace in order to place the user in the best context[4].

Awareness is primordial for collaborative modeling. Nevertheless, awareness is little supported in commercial tools and research work. Therefore, we decided to do a study on this aspect. As awareness is centered on the needs of users, we decided to follow the user-centered design process. Our study is the first step of the process. The goal of our study is to know how modeling tools users work in collaboration. We want to know what the different articulation work types are and what awareness information is needed in these steps. To complete this study, we used GenMyModel and the many collaboratives projects constructed by users and we analyzed them. With this analysis, we determined three articulation work types. Then, for each articulation work type, we surveyed users of modeling tools to evaluate the importance of each awareness information. While we mainly contacted GenMyModel users for the survey, we also included users of other modeling tools in order to generalize our results: we took care to assess whether the modeling tool used influences answers. To go further with our surveys, we researched if correlations between distance between users, the size of the project, the size of the team or the context of the project, and awareness needs exist.

The remainder of the paper is organized as follows. In section 2, we will describe awareness in depth, what are awareness information types and how they are currently handled by commercial modeling tools and research works. In section 3, we will present the study with the analysis of collaborative projects to define different articulation work types. In section 4, we will link articulation work types with awareness information. From these links, we will look at the surveys for modeling tool users in section 5. Finally, we will present the results in section 6.

2 Awareness

In this section, we will present the CSCW dimension which mainly guides our experiment: awareness. Then, we will see how awareness is supported in modeling tools.

2.1 Global Definition

The most widely used definition of awareness is given by Dourish and Bellotti [4] : “[...] an understanding of the activities of others, which provides a context for your own activities”. Schlichter [5] built on this aspect of CSCW a few years later in order to reach and to maintain effective coordination of collaborative work. In his review of awareness in Distributed Collaborative Software Engineering [6], Omoronyia defines four types of awareness as follows:

Workspace Awareness. The up-to-minute knowledge of other participants interactions with the shared workspace[7].

Informal Awareness. The general sense of who is around, what they are doing, and what they are going to do[8].

Group-Structural Awareness. Knowledge about people roles and responsibilities, their positions on an issue, their status, and group processes[8].

Social Awareness. Information about the presence and activities of people in a shared environment[9].

These types refer to a series of questions people ask themselves when they collaborate. Table 1 shows a list of the main ones.

Workspace awareness is the most studied in awareness literature [6], and it seems to also be the most important type in Distributed Collaborative Software Engineering. Even if awareness is the focus of many scientific works for decades, it is still a scientific issue. The April 2013 volume of Computer Supported Cooperative Work [11] presents recent scientific results on awareness in different areas (distributed software development, model versioning, virtual team, social media, etc.). Additionally, the new direction of research relates to recent interactive technologies like large tactile surfaces. Yuill [12] reports case studies about awareness with such technologies.

In this part, we presented awareness information for collaboration. We must now know how awareness is supported in modeling tools.

2.2 Awareness in Modeling

In the previous part, we saw that awareness is important for collaboration and awareness involves displaying a lot of information. Now, we would like to know what modeling tools provide collaborative features, how they provide this collaboration and what awareness information they display. Few works exist about awareness in modeling tools. Both industrial tools and research prototypes do not implement collaboration in the same way. On the one hand, users of the

Table 1. Awareness answers to collaboration raised questions [10]

Awareness dimension	Element	Question
Workspace	Identity	Who is that?
	Authorship	Who is doing that?
	Action	What are they doing?
	Action history	How did that operation happen?
	Intention	What goal is that action part of?
	Intention history	What goal was that action part of?
	Artifact	What object are they working on?
	Artifact history	How did this artifact come to be in this state?
	Location	Where are they working?
	Location history	Where has a person been?
	Gaze	Where are they looking?
	View	Where can they see?
	Reach	Where can they reach?
	Event history	When did that event happen?
Informal	Opinion	What is their opinion?
	Presence	Is anyone in the workspace?
	Presence history	Who was here and when?
Social	Interest level	How interested are they?
	Emotional feelings	How are they feeling?
	Availability	Are they available?
Group-structural	Roles and responsibilities	What are their roles/positions?

tool share the same environment and edit the same version of the model. On the other hand, users have to retrieve the version of the model locally, modify it and commit and optionally merge the result if users have modified the same elements. Furthermore, in this study, we did not differentiate modeling tools and diagramming software. The difference between both tool types is that the model made with a modeling tool is valid. With modeling tools, many actions are forbidden for compliance with model specifications contrary to diagramming software. Most desktop modeling tools do not appear in our presentation of modeling tools. They provide a classic revision control system for collaboration. Visio provides the best awareness support, so we decided to present only Visio among the desktop modeling tools.

Commercial Tools. Concerning commercial modeling tools, Gliffy² is a web-based tool and provides a desynchronized environment. Each collaborator has a local version of the model. Collaborators do not know if others are editing the model or on what they are working. Awareness information is missing.

Contrary to Gliffy, Visio³ is a desktop tool but also provides desynchronized collaboration. Until the user saves the model, the user can not see changes done by collaborators. When users are editing the model, they know the number of connected collaborators and their names. If they work on elements close to those

² <https://www.gliffy.com>

³ <http://office.microsoft.com/en-001/visio/>

on which their collaborators are working, they are notified that a collaborator has made changes on this element. It involves support of *Identity*, *Presence*, *Authorship partially* and *Artifact partially* awareness elements. *Authorship* and *Artifact* support is partial because if the user is not working on the same diagram as your collaborators, the user can not know who has changed the model.

Like Gliffy, Creately⁴ is a web-based tool. Creately provides real-time collaboration like Visio. Available features to display awareness information are the list of connected users with their names, a focus on what elements are selected by collaborators, and comments on model parts. The features concern only real-time information and related awareness elements are *Identity*, *Authorship partially*, *Artifact partially* and *Presence*. *Authorship* and *Artifact* support is partial for the same reason as in Gliffy.

Finally, the tool which has the largest support for awareness is Lucidchart⁵ which is a web-based tool with real-time collaboration. Lucidchart provides a chat panel where collaborators know who is connected and their names. Users can add comments on model elements in order to assign tasks to themselves or to collaborators. Nevertheless, we do not know if a user is working on a specific task when the user is editing the model. Lucidchart also allows users to revert the model to a revision, but users can not see differences between two versions. Thus, Lucidchart supports many awareness elements: *Identity*, *Authorship*, *Intention partially*, *Intention history*, *Artifact partially*, *Event History* and *Presence*. Thanks to the revision history, users can know who has changed the model, but they can not see what model elements have changed.

Research Prototypes. The main research works on collaboration in modeling tools are centered on collaboration protocol, model merging, etc. but few deal with awareness and are centered on users. Both of the following research prototypes are web-based and provide real-time modeling.

The first is GEMSjax (a meta-modeling tool)[13]. In his works, Farwick briefly discusses awareness. He summarizes it as a chat and events to see which elements are currently selected but without giving more details. GEMSjax supports *Authorship partially* and *Artifact partially* information.

Thum with his work on SLIM [14] is akin to Farwick's work on the point of view of awareness. SLIM is a UML Case Tool for synchronous collaborative modeling based on a lock system to avoid conflicts. When a user clicks on an element, a lock is put in place to prevent modifications by another user. Concerning awareness, SLIM contains a chat to discuss, displays current connected users, and padlocks on model elements locked by other users . With these features, SLIM supports *Identity*, *Presence*, *Authorship partially* and *Artifact partially*.

Both tools support *Authorship* and *Artifact* partially, because if users work on different diagrams, they lose information since they can not see changes.

⁴ <http://creately.com/>

⁵ <https://www.lucidchart.com/>

Other works exist on awareness but they deal more with the organization of awareness information. Gallardo [15] provides an ontology to organize workspace awareness information for modeling tools.

Issues. Awareness is a crucial aspect for collaborative work. Nevertheless, tools and research prototypes do not seem to be interested in awareness. A quick look to widespread collaborative authoring tools like Google Drive Applications or Office Web Apps shows that they provide a wider support of awareness: *Authorship*, *Identity*, *Location*, *Artifact/Action*, *Artifact/Action History* and *Presence*. Unfortunately, it seems difficult to assess whether the additional awareness features need to be applied in the modeling context without a thorough investigation.

The problem with tools and research prototypes is the multiplication of widgets in order to display information. However, as display area is limited, it is not possible to have more and more widgets to cover awareness elements. Furthermore, users risk a cognitive overload due to the amount of information. Therefore, we have to know what information is required the most. Then, we have to see if requirements vary not only according to different articulation work types but also according to the size of the team, the size of the project, the type of project which is modeled, the project's context or the distance between collaborators.

To complete these objectives, we have questioned people who have already used collaborative modeling through web surveys. These surveys determined what awareness information is relevant for the user. As we think that the relevance of awareness information varies according to articulation work types, first, we studied what these different articulation work types are. These articulation work types are presented in section 3. We will see that results on articulation work types allow us to determine what information is relevant for each type in section 4. In section 5, we will detail how we have built surveys for each articulation work type. Section 6 will present the results and discussions around them.

In future works, with these requirements, we will be able to implement better software support and also evolve the requirements to improve our knowledge on what is needed for collaborative modeling. We can note that our approach is user-centered with analysis, implementation and tests. This approach seems to be adapted because awareness is mainly a HCI issue.

3 Defining Articulation Work Types

The first step of our approach is to define different articulation work types. To do this, we analyzed GenMyModel user projects. In order to analyze GenMyModel user projects, we analyzed the database. We extracted collaborative projects with the following criteria:

- Public, to have the right to analyze the project's content.
- Not cloned from GenMyModel examples to avoid tests projects. GenMyModel examples can be cloned in order to discover GenMyModel's capabilities.
- Recent, in order to contact users about a project they remember.

From March to mid-May, 508 projects met these criteria. For each project, we built users' activity sessions by analyzing launched actions on their projects as shown in figure 1. With these sessions, we can analyze how users interact with the model, and if they edit the model at the same time. For example, in figure 1, we can see numbers in rectangles. Each rectangle corresponds to a user session where actions were launched, and the number indicates the number of launched actions. We can see that the figure is divided into two parts. In the first, the user is the only one to make changes. In the second, user *thampe* initialized the project with 9 actions. Then, 2 collaborators joined him on the project, and they started to edit the model at the same time. They stopped for 10 minutes and started again until user *angart* is alone to modify the model. After *angart* stopped modifying the model, user *thampe* took the relay and modified the model by himself.

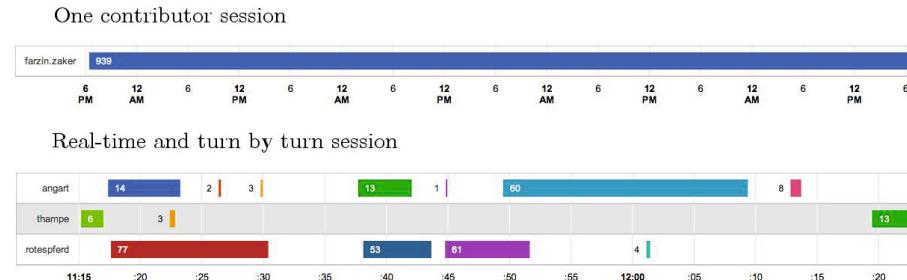


Fig. 1. Examples of sessions

From the project analysis, we determined three articulation work types.

One contributor articulation work (OC): One user modifies the model and the others are spectators. The spectators never interact with the model while the one user is the author of all actions. 308 of the 508 selected projects which satisfied the aforementioned criteria have only one contributor.

Users modify the model turn by turn: (TBT) One user modifies the model and stops modifying. After this, another user does many actions and then stops. The time between a stop and the start of following changes can be one day, a few hours or a few minutes. For example, in a project, we have seen three users take turns every two minutes.

Users modify the model at the same time (RT): More than one user modifies the model in real-time. Real-time collaboration is little used. It represents only 5% of collaboration time. Nevertheless, this time could be strategic for the coordination between users. We can not know if RT sessions have influenced the following work. Moreover, the utility of RT was not our first objective.

Often, turn by turn and real-time articulation work types are used in the same project. We defined three different articulation work types, and now we have to link awareness information requirements with them.

4 Linking Articulation Work Types and Awareness

In this section, we will discuss what awareness information may be relevant for each articulation work types.

4.1 One Contributor

A plausible situation is that the only one contributor is an architect and others are the developers. Developers have to know when parts of the model are finished in order to code them. Furthermore, developers can give their ideas on model improvements. For this articulation work type, the relevant awareness elements are *intention* and *opinion*.

4.2 Users Modify Model Turn by Turn

Most of the awareness elements are from the past, because no other user has had interaction with the shared environment. The user needs information of what changes took place to understand the actual state of the model. Thus, users may have to know what the collaborators' current tasks are in order not to modify the same model parts. For this articulation work type, the relevant awareness elements are *authorship*, *action history*, *intention*, *intention history*, *artifact history*, *location history*, *event history*, and *presence history*. During the analysis of the users activity sessions, we observed another important element. In fact, the time between login and the first command is superior than 1 minute. There are two possible reasons: it is the time needed for users to see all the changes since the last disconnection, or the users look at another browser tab while their model is loading. If the reason is the time needed to see changes since last connection, it means that past events are at least relevant when they log in to their projects.

4.3 Users Modify the Model at the Same Time

All awareness information may be needed for real-time collaboration, because the information about current actions, collaborators, past actions, etc. could be required in order to understand the current state of the model.

Table 2 summarizes the related awareness elements according to the articulation work types.

Table 2. Possible awareness elements used in articulation work types

Element	OC	TBT	RT
Identity			X
Authorship		X	X
Action			X
Action history		X	X
Intention	X	X	X
Intention history		X	X
Artifact			X
Artifact history		X	X
Location			X
Location history		X	X
Gaze			X
View			X
Reach			X
Event history		X	X
Opinion	X	X	X
Presence			X
Presence history		X	X
Interest level			X
Emotional feelings			X
Availability			X
Roles and responsibilities			X

From the table 2, we defined questions to sort awareness elements by relevance for each articulation work type.

5 Developing Web Surveys

In the previous section, we defined what awareness elements are concerned according to the articulation work types. Therefore, we discussed with GenMyModel users to define their requirements. To do this, we developed surveys to send to GenMyModel users who had projects which satisfied the aforementioned criteria. Moreover, we shared our surveys on social networks according to the articulation work types. We did this in order to have responses from people who do not use GenMyModel and to know if the use of GenMyModel influences results. Now, we will describe these surveys. For each, we asked them the following information:

- Project context (student project, business project, research project)
- Modeled application type (users model for Desktop, Web or Mobile application)
- Team size (1-3, 3-5, 5-10, 10+)
- How far away are you from your collaborators? (We share the same building, town, country, we are over the world)

Each question corresponds to a potential element which can be correlated to the awareness requirements. We will have to verify further if the correlations found impact the awareness requirements. Now, we will detail the different surveys according to the articulation work type.

5.1 One Contributor

Aforementioned possible awareness elements when collaborative projects have just one contributor are *intention* and *opinion*. For this articulation work type, we contacted the contributors of 308 projects by email, and we asked the following questions:

- Do you need the opinion of your collaborators?
- If yes, in what context?
- How do you ask their opinion?
- Do others know what your current task is?
- if yes, by what way do they have your current task? How do they use this information?

The first, the second and the third questions determine if opinion is important or not for users and the way to know the opinion of their collaborators. The last questions target intentions. The goal of these questions is not only to know if *Opinion* and *Intention* are important for users but also to know how they obtain and share this information.

5.2 Users Modify Model Turn by Turn and at the Same Time

As most of the collaborative projects with many contributors use TBT and RT articulation work types, we defined one survey divided into two parts that concerned each articulation work type. In the previous section, we linked the articulation work types with awareness information. Thus, for each awareness information, we had to evaluate the information requirement. We asked the usefulness of awareness information on a scale from 1 to 4 where 1 is useless and 4 is useful. This pair scale forced users to have an opinion on the question. They could not be neutral. To simplify the survey, we generalized awareness information to avoid explaining the sense of each.

For example with RT, for *Presence* information, we asked the usefulness of *Knowing if other collaborators are connected*, and for *Identity*, we asked if the *Name of connected collaborators* is relevant.

Furthermore, we asked them to give the 5 most important elements aforementioned in order to classify awareness information by relevance. The evaluation of each information on the scale is complementary to the classification, because the evaluation is useful to detect inconsistencies in responses.

In the second part, with the same type of scale, the respondents evaluated the usefulness of information for TBT. This is when the user works alone on the project. As for RT, we simplified the surveys and instead of asking the usefulness of *Presence History*, we asked the relevance of *Who was connected when you were offline*.

Moreover, we asked the question *When you open the model, what is the first thing you do?* in order to understand why the time between login and the first command is more than one minute. After the questions, we provided a section for comments so that users could offer their ideas about collaboration. For this survey, we contacted 392 collaborators from 508 projects which satisfied the aforementioned criteria. In this email, we asked the person if they wanted to help us to improve collaboration features, and we waited for their responses. It was only to open the discussion. When users were ready to help us, we sent them the online survey made with Google Form. We also posted this form on LinkedIn UML professionals groups in order to have responses from people who do not use GenMyModel. In sharing the survey with others, we wanted to know if the tool influences responses.

6 Results

The following results are presented by articulation work types.

6.1 Ranking Awareness Information by Relevance

One Contributor. For this first survey, we had 17 responses from the 308 contacted users.

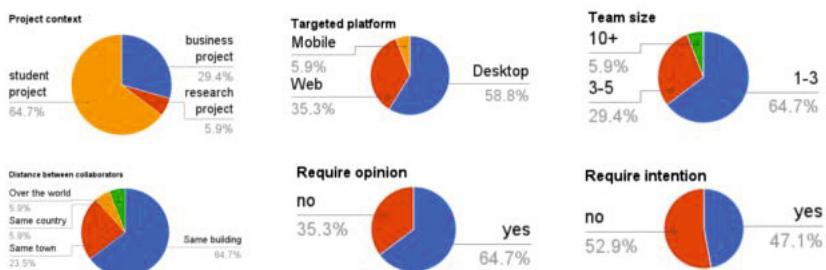


Fig. 2. Survey responses for OC

Figure 2 summarizes the results of the first survey. We learned that it is mostly students who use the one contributor articulation work type. Therefore, most of the respondents model for desktop applications. Most of the respondents work in little groups, and most of them share the same building. Most respondents need the opinion of their collaborators even if they do not make modifications on the model. For the question, *Why do you need the opinion of your collaborators?*, they answered that they wanted the advice of others in order to improve the quality of the model.

Concerning intentions, the respondents are more divided. Less than half shared their tasks with others. For these users, all tasks were written in a product and issue tracking product, in a Facebook group or were defined during meetings.

Users Modify Model Turn by Turn. With this survey, we had 50 responses, 35 are GenMyModel users contacted by email, and 15 others came from LinkedIn professional groups for modeling. As figure 3 shows, the three most important awareness elements when users modify the model turn by turn are *Action History/Artifact History*, *Authorship* and *Intention*.

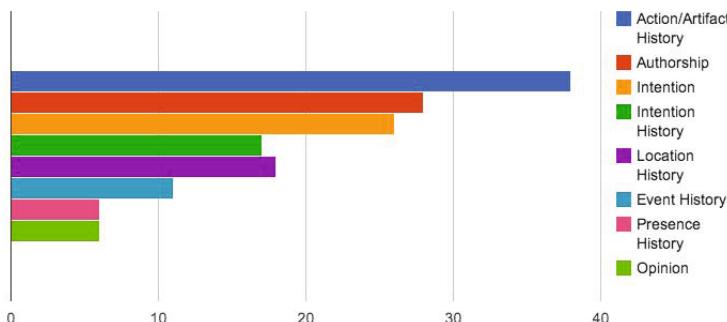


Fig. 3. Ranking of responses for TBT

Users Modify the Model at the Same Time. Figure 4 shows the ranking of each element defined as the most important by users. The most required awareness information for RT is *Presence*, *Identity*, *Action/Artifact*, *Intention* and *Authorship*.

Discussions. If we compare the required awareness elements and the existing information in modeling tools, we can see there is a gap. For OC, even if the number of respondents is low, responses give some cues about how they collaborate with only one contributor. On the one hand, modeling tools have to provide support with issue tracking tools like Jira, Mantis or their own solution, because half of the users share their tasks with their collaborators. On the other

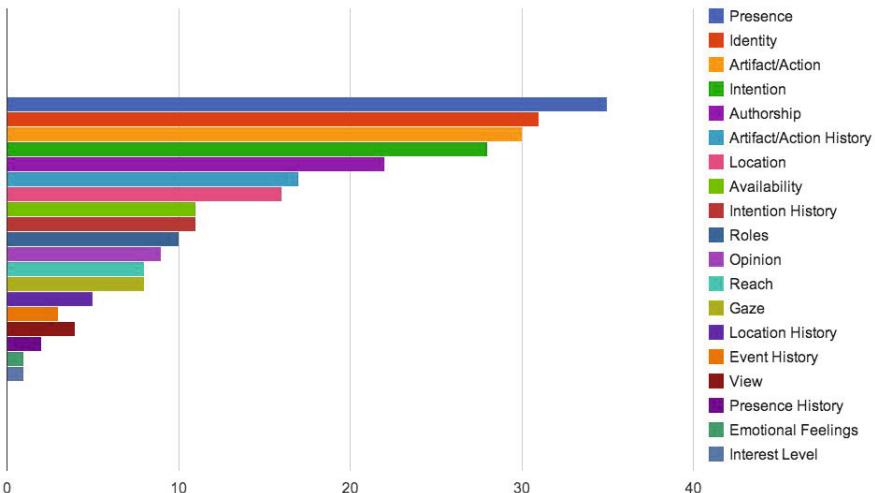


Fig. 4. Ranking of responses for real-time collaboration

hand, as opinion is important, modeling tools have to find a way to retrieve the opinion of collaborators. Currently, the modeling tool with the largest support of awareness is Lucid Chart which provides task features. Nevertheless, Lucid Chart does not allow users to obtain the opinion of collaborators. It is also interesting to mention that tasks and opinions are missing in collaborative authoring tools.

For TBT, the most important awareness information is Artifact/Action History. This information is not covered in modeling tools and research prototypes. As users work alone in TBT, they do not need information about the current changes or their collaborators. Thus, it is logical to have information concerning Artifact/Action History first. The second most important information is related to the first. Respondents want to know the past changes, and furthermore, they want to know who has made these changes. Finally, users want to know what is the current tasks of their collaborators. It can be easily explained. Users do not want to modify model parts which can be modified by others. This information is the only one which is provided by Lucid Chart, but only partially, because it is not possible to know if users are working on tasks. We can see that modeling tools provide a little support for TBT. Even research works in collaborative modeling do not cover the most important awareness information for TBT. In general, the ontology of Gallardo does not refer to past events. Concerning collaborative authoring tools, their awareness support seems fit to TBT as the two most requested information are provided. However, tasks are still missing.

Concerning RT, the most important awareness elements are *Presence*, *Identity*, *Action/Artifact*, *Intention* and *Authorship*. As changes are not related to tasks, and users can not follow changes on another diagram, Lucid Chart does

not cover *Action* and covers partially *Intention* and *Authorship*. As for TBT, collaborative authoring tools do not provide information about tasks.

At the beginning of our study, we observed that the time between login and the first modification is more than one minute. The first thing users do when they log in to the project is to look at the last changes since their last disconnection. Nevertheless, the past awareness information does not appear in the most required information defined by users. It means that they need this information only when they log in to be aware of the project events since their last disconnection.

In section 2, we said that workspace awareness is the most studied in awareness literature. The importance of workspace awareness is confirmed for modeling tools with our study. Most of the important awareness information in the ranking is workspace awareness information.

6.2 Correlations with the Awareness Requirements

For each articulation work type, we researched if the size of the team, the size of the project, the modeled application type, the context of the project, the distance between collaborators or if being a GenMyModel user (named elements in the rest of the article) were correlated with awareness information requirements. To do this, we tested the correlation with responses where respondents had to answer on a scale from 1 to 4. To complete the correlation test, we used a Pearson Correlation test with the SPSSStatistics tool. This test gives correlations between 2 variables. To begin with the tests, we verified that elements were independents. As elements were independents, we verified if they change awareness requirements for each articulation work type.

One Contributor. The Pearson Correlation tests did not reveal links with *intention* or *opinion* awareness information. As the number of respondents is low, it is not surprising. It could be interesting to go further with this study.

Users Modify Model Turn by Turn. We present the correlations with awareness requirements for TBT. All of them are resumed in figure 5. In this figure, solid arrows show that the awareness information requirement increases with the element. The values inside brackets mean that the requirement increases with the conditions represented by these values. Significance levels are indicated with stars (** for 0.01, ** for 0.05 and * for 0.10).

For example, in the figure, we can see that *opinion* requirement increases with the size of the team. We can also see that *Presence History* information is more important for users who model a desktop application.

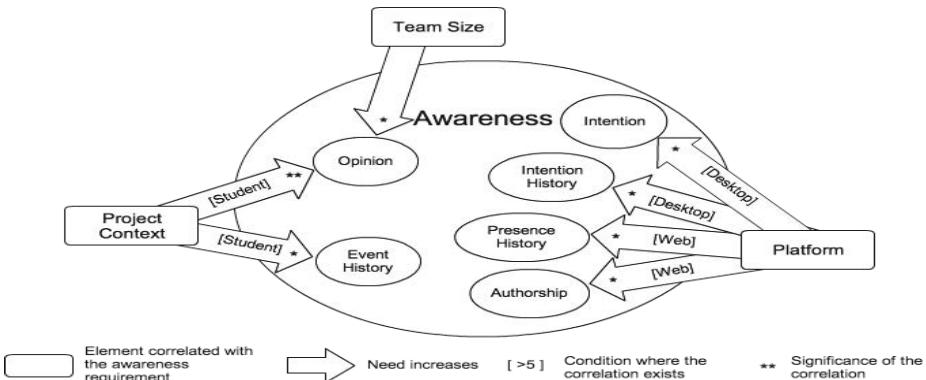


Fig. 5. Correlation between parameters and awareness elements when TBT

Users Modify the Model at the Same Time. We present the correlations with awareness information requirements for RT. Figure 6 shows these correlations. In this figure, dotted arrows mean that the awareness information requirement decreases with the associated element.

Discussions. For TBT, the *opinion* requirement is higher when the size of the team increases. We are not able to explain why and we will have to discuss this with modeling tool users to understand. In the case of a student project, requirements for *Opinion* and *Event History* are more important. It is not surprising, because students are not modeling experts. They need more information from others to build their models together. If modeling tools asked at the creation of the project what the type of project it is, it could be possible to customize displayed awareness information. With this option, users would only see the information important to their projects without useless information. Furthermore, for *Intention History* information which is at the fifth position in the ranking, the need increases when users model for a desktop application. Currently, we do not have an explanation about the correlation between modeled application type and awareness information. We have to examine user behavior in depth to understand this correlation. Nevertheless, in the case of users who model a desktop application, the ranking could be a little different.

For RT, the size of the team and the size of the project are widely correlated with the awareness information requirements. When the size of the project increases, users have more of a need to know *intentions* and *changes done* on the model. It is not surprising, because they have to organize themselves for a large project and know the last changes to understand the current state of the model. As *intention* and *past changes* on the model are already well ranked, it is very important to highlight this information when the project is large. The larger the project becomes, the more important this information is. As for TBT, *opinion* requirement increases with the size of the team.

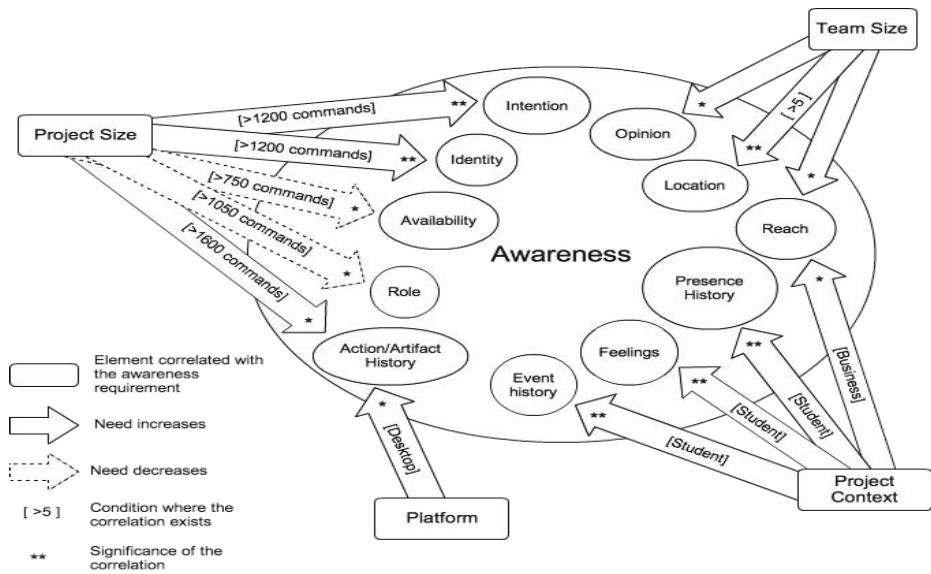


Fig. 6. Correlation between parameters and awareness elements for RT

As the tool used by the survey respondents is not correlated with the results, the rankings and the correlations are applicable for all modeling tools.

Despite correlations exist with the awareness information requirement, modeling tools and research work do not vary the display of awareness information. The same observation is valid for collaborative authoring tools. We will have to realize other studies with modeling tool users to understand why the modeled application type is correlated with awareness information requirements. With these studies, we could refine ranking according to the aforementioned correlations.

Furthermore, in this study, we did not test the correlations of multiple elements at the same time, it will be the topic of another study.

7 Conclusion

In this paper, we have seen that GenMyModel was developed to counteract installation problems which can be time-consuming, and the fact that users have to ensure they use the same version of the modeling tool to share their models. As CSCW is user-centered, particularly awareness, we focused our research on this dimension. We looked at awareness support in commercial modeling tools and research prototypes. They provided a little support of awareness. We decided to launch a study with modeling tool users by following the user-centred design process. To begin the study, we analyzed user projects built with GenMyModel in order to categorize how users collaborate, and we defined three articulation work

types: projects with only one contributor and projects with many contributors, divided into two categories. On the one hand, we have projects which use a real-time collaboration feature, on the other hand, projects where users edit the model turn by turn. For each articulation work type, we defined what awareness elements could be needed for users.

After this analysis, we have the following contributions. The first is the ranking of awareness elements according to the articulation work type. The ranking is different for each. First, for projects which have only one contributor, the awareness elements required are *intention* and *opinion*. Then, for the turn by turn articulation work type, the most important awareness information is *artifact/action history*, *authorship* and *intention*. To finish, the real-time articulation work type requires the support of *presence*, *identity*, *artifact/action*, *intention* and *authorship*.

The second contribution is the correlations between elements and the awareness requirements. In fact, for RT, the size of the project, the size of the team, the modeled application type and the context of the project increase/decrease the *Intention*, *Identity*, *Availability*, *Role*, *Artifact/Action History*, *Opinion*, *Location*, *Reach*, *Presence History*, *Feelings* and *Event History* requirements. For TBT, the requirements are correlated with the size of the team, the context of the project and the modeled application type. The correlated requirements are *Opinion*, *Event History*, *Intention*, *Intention History*, *Presence History* and *Authorship*. We will have to verify these correlations when we implement the awareness information. We can then verify if the awareness requirements change according to aforementioned elements. Except *Opinion* and *Intention*, all required awareness features for collaborative modeling have already been proposed by collaborative authoring tools. It would be interesting to study to what extent these features may catch the interest of users of these tools.

As it is not possible to display too much information due to a limited display space, and in order to prevent the cognitive overload, we will study if navigation is possible between the most important awareness elements. For example, when the collaborators' identities are displayed, what possible awareness elements could be displayed next to the users?

First, we will have to define a navigation system for awareness information, and then, we will find out how to display this information. To do this, we will follow the next two steps of the user-centered design process, implementation and tests.

References

1. Bannon, L.J., Schmidt, K.: Studies in computer supported cooperative work, pp. 3–16. North-Holland Publishing Co., Amsterdam (1991)
2. Schmidt, K., Bannon, L.: Taking cscw seriously. Computer Supported Cooperative Work (CSCW) 1(1-2), 7–40 (1992)
3. Herrmann, T., Nolte, A., Prilla, M.: Awareness support for combining individual and collaborative process design in co-located meetings. Computer Supported Cooperative Work (CSCW) 22(2-3), 241–270 (2013)

4. Dourish, P., Bellotti, V.: Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work, CSCW 1992, pp. 107–114. ACM, New York (1992)
5. Schlichter, J., Koch, M., Xu, C.: Awareness - the common link between groupware and community support systems. In: Ishida, T. (ed.) Community Computing and Support Systems. LNCS, vol. 1519, pp. 77–93. Springer, Heidelberg (1998)
6. Omoronyia, I., Ferguson, J., Roper, M., Wood, M.: A review of awareness in distributed collaborative software engineering. *Software: Practice and Experience* 40(12), 1107–1133 (2010)
7. Gutwin, C., Stark, G., Greenberg, S.: Support for workspace awareness in educational groupware. In: The First International Conference on Computer Support for Collaborative Learning, CSCL 1995, pp. 147–156. L. Erlbaum Associates Inc., Hillsdale (1995)
8. Gutwin, C., Greenberg, S., Roseman, M.: Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. In: Proceedings of HCI on People and Computers XI, HCI 1996, pp. 281–298. Springer, London (1996)
9. Prinz, W.: Nessie: An awareness environment for cooperative settings. In: Bødker, S., Kyng, M., Schmidt, K. (eds.) ECSCW 1999, pp. 391–410. Springer Netherlands (1999)
10. Steinmacher, I., Chaves, A., Gerosa, M.: Awareness support in distributed software development: A systematic review and mapping of the literature. *Computer Supported Cooperative Work (CSCW)* 22(2-3), 113–158 (2013)
11. Kolfschoten, G., Herrmann, T., Lukosch, S.: Differentiated awareness-support in computer supported collaborative work. *Computer Supported Cooperative Work (CSCW)* 22(2-3), 107–112 (2013)
12. Yuill, N., Rogers, Y.: Mechanisms for collaboration: A design and evaluation framework for multi-user interfaces. *ACM Trans. Comput.* 19(1), 1:1–1:25 (2012)
13. Farwick, M., Agreiter, B., White, J., Forster, S., Lanzanasto, N., Breu, R.: A web-based collaborative metamodelling environment with secure remote model access. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 278–291. Springer, Heidelberg (2010)
14. Thum, C., Schwind, M., Schader, M.: Slim—a lightweight environment for synchronous collaborative modeling. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 137–151. Springer, Heidelberg (2009)
15. Gallardo, J., Molina, A.I., Bravo, C., Redondo, M.A., Collazos, C.A.: An ontological conceptualization approach for awareness in domain-independent collaborative modeling systems: Application to a model-driven development method. *Expert Systems with Applications* 38(2), 1099–1118 (2011); *Intelligent Collaboration and Design*.

Trusted Dynamic Storage for Dunbar-Based P2P Online Social Networks

Marco Conti², Andrea De Salve¹, Barbara Guidi^{1,2},
Francesco Pitti¹, and Laura Ricci¹

¹ University of Pisa - Department of Computer Science,
Largo B. Pontecorvo, 56127, Pisa, Italy

² IIT-CNR, via G. Moruzzi, 1 56124 Pisa, Italy
`{desalve, guidi, pittof, ricci}@di.unipi.it,`
`m.conti@iit.cnr.it`

Abstract. Online Social Networks (OSNs) are becoming more and more popular in today's Internet. Distributed Online Social Networks (DOSNs), are OSNs which do not exploit a central server for storing users' data and enable users to have more control on their profile content, ensuring a higher level of privacy. The main challenge of DOSNs comes from guaranteeing availability of the data when the data owner is offline. In this paper we propose a new P2P dynamic approach to the problem of data persistence in DOSNs. By following Dunbar's approach, our system stores the data of a user only on a restricted number of friends which have regular contacts with him/her. Users in this set are chosen by considering several criteria targeting different goals. Differently from other approaches, nodes chosen to keep data replicas are not statically defined but dynamically change according to users' churn. Our dynamic friend selection achieves availability higher than 90% with a maximum of 2 online profile replicas at a time for users with at least 40 friends. By using real Facebook data traces we prove that our approach offers high availability even when the online time of users is low.

Keywords: DOSN, P2P, Data availability, Dunbar.

1 Introduction

In the last few years Online Social Networks (OSNs) have achieved unprecedented success, changing the way of how people interact with each other and becoming storehouses of huge amount of data in the form of messages, photos and personal information. This information, on one hand, is of great business value to the service provider, but on the other hand exposes the users to the risk of privacy breaches and malicious exploitation. Users who subscribe to a centralized OSN service have to agree the policies of the service provider permitting it to share their personal information with third-parties and loosing any form of control on their data.

A current trend for developing OSN services is towards the distribution of the social network infrastructure, often through a P2P approach. A Distributed

Online Social Network (DOSNs) [1] is an OSN implemented on a distributed information management platform, such as a network of trusted servers, P2P systems or opportunistic networks.

In a P2P Distributed Online Social Network there is no single provider but a set of peers that take on and share the tasks needed to run the system. The development of the existing functionalities of OSNs in a distributed context requires finding ways for providing robustness against churn, distributing storage of data, propagating updates, defining an overlay topology and a protocol enabling searching and addressing, etc. One of the main challenge comes from guaranteeing the data persistence of a user data even when he is offline. Some DOSNs rely on external storage systems, for example exploiting a distributed file system [2], while some other more recent approaches ([3,4,5]) propose to store social data of a user on the storage support provided by users' friends. In these proposals, the data owner serves his own data when he is online, and elects a subset of his friends to make the data available when he is offline.

The focus of the paper is on the problem of data availability. Since users can connect to/disconnect from the system, the social overlay is dynamic. Moreover, new friendship relationships may be formed or old ones severed but since such changes are infrequent, we disregard them in our analysis. Our goal is guaranteeing there is always a copy of the profile on an online node for each node in the network. First, we introduce our P2P Distributed Online Social Network based on the Dunbar approach. Then, since privacy is a major problem in DOSNs, we exploit the information returned by the analysis of the social interactions among the users to place profile replicas on trusted users.

We have evaluated the system through a set of simulations conducted on real Facebook traces which show that a user has almost always at least one online friend. These experimental results show the soundness of our approach.

The rest of this paper is structured as follows. Section 2 introduces the model of social network we refer to. The system we propose is described in Section 3, while Section 4 introduces the strategies we have defined to choose the Point of Storages, i.e. the friends storing the user's profile. The algorithms for the election of these nodes are presented in Section 5. Section 6 presents the methodology exploited to conduct the experiments, while Section 7 shows the experimental results. Finally, related work is reported in Section 8 and Section 9 presents the conclusions.

2 The Context

A social network is formally described by an undirected graph $G = (V, E)$, where vertexes in V correspond to users and edges in E to the social relationships between them.

An interesting concept which has been recently introduced is that of *ego network*. Everett and Borgatti in [6] define ego networks to be networks consisting of a single actor (*ego*) together with the actors they are connected to (*alters*) and all the links among those alters. Thus we can formally define the ego network of

a user u as $EN(u) = (V_u, E_u)$, where $V_u = \{u\} \cup \{v \in V \mid (u, v) \in E\} \wedge E_u = \{(a, b) \in E \mid a = u \vee b = u \vee \{a, b\} \subseteq V_u\}$. The ego network of a user can be seen as a subset of G which represents the local view of the user.

The size of an ego network may be large, for instance the maximum amount of relationships a user can establish is generally very high (5000 connections in Facebook). [7] and [8] show that the number of friends a user can maintain stable social connections with is approximately 150. This limit is known as *Dunbar's number*. The stability of a connection may be defined as a function of the *tie strength* which characterizes the relation and is a numeric value that quantifies the strength of the relationship between two users. In OSNs, the tie strength can be calculated by taking into account different factors such as contact frequency between the two users, the number of likes, posts, comments, private messages, tags, etc. [9].

Our intuition is to use this limit to create a dynamic social overlay where each user is connected only to a reduced number of friends arranged in a hierarchical inclusive sequence ordered by increasing level of intimacy [10]. This permits us to reduce the total amount of social information each peer has to keep in memory and the number of connections of the P2P overlay. In our model each user can have at most 150 friends in his ego network, which are the nodes with whom the user has the strongest relationships in terms of tie strength. In the following, this ego network will be referred as *Dunbar-based ego network*.

In the following, we use $ef(u)$ to represent the set of all the friends of a user u belonging to his ego network and $df(u)$ to represent the set of his friends that belong to his Dunbar based ego network. We call these friends *Dunbar ego friends*. Nevertheless, since our ego network structure is dynamic, a node belonging to $ef(u) - df(u)$ can become part of $df(u)$ after a high number of contacts with u and viceversa.

Since we define a bijective mapping between users in the social network and P2P nodes, in the following we will refer to users and nodes interchangeably. We further assume that each user uses exactly one device, i.e. we do not consider the case of one user using multiple devices nor the case of multiple users sharing the same device.

2.1 Profile-Based Communication

In a DOSN each user is associated with his *profile*, which is a personal web page where users freely post content – e.g. text, snippets, pictures, videos and music. In response to these postings other users, usually friends, post comments and other content. In modern OSNs communication is said to be *profile-based*, since around 90% of server requests are related to profile page content [11]. Profiles generally have *small* dimension, since pretty much everything in the profile pages – comments, links, thumbnails, small pictures – are small objects. The only exception are movies and large collections of high-quality pictures: however, these are usually not part of profiles anyway, and are linked from services such as YouTube and Flickr [12]. Since profile pages are generally small we assume that the time to replicate a user's profile on one of his friends is negligible [4].

3 The System Model

Our system grounds on a P2P overlay where the connections between nodes correspond to the social relations of the Dunbar-based ego networks of the users. Furthermore, a DHT [13] is exploited for peer bootstrapping, for supporting the search of new friends and for other management tasks. The description of the DHT is not the main purpose of this paper, hence in the following we will discuss only how it is exploited by the distributed storage support.

3.1 Dynamism and Points of Storage (PoS)

DOSNs should guarantee that the copy of the profile of each user is available at any time. We manage the problem of data availability by introducing a trusted replication approach in which each user dynamically elects a subset of his Dunbar friends, which we will refer as Points of Storage (PoS), to store a replica of his social data. We define $Prof(u)$ as the profile of a node u , $PoS(u)$ as the set of Point of Storages of node u and $PoS^{-1}(u) = \{n \mid u \in PoS(n)\}$ as the set of nodes for which u is Point of Storage. The PoS of a user serves his data when he is offline. Each PoS elects another node as PoS if it disconnects from the system, and this election may be recursively executed. As long as there is at least one online friend for each node the availability can be always satisfied but if no friend is online, then data stored in the system will not be accessible by any means. As long as no friend of the user is online that is a minor turn-off, since no one is interested in accessing the user's profile. But if a friend of the user reconnects to the system and does not keep in memory a copy of the user's profile he cannot access the user's data until one PoS of the user or the user himself reconnects to the system. To reduce the effect of this situation which, as we will discuss in Section 7, happens very rarely, our system behaves as follows:

- each user u that is going to disconnect from the system passes a copy of the profile of each node $n \in PoS^{-1}(u)$ to a node in $ef(n)$ but keeps a (possibly) outdated version of the profile in memory until his reconnection to the system;
- when a user u reconnects to the system he destroys his local copy of the profile of each friend v unless there is no online copy of that profile. In the latter case u notifies on the DHT it is the current PoS for v and provides a (possibly) outdated copy of $Prof(v)$ in the system;
- the DHT, which we refer to as *PoS Table*, stores, for each user u , a list of entries of type $< PoS(u), timestamp, IsOnline, IsOutdated >$ where $PoS(u)$ is the identifier of a node that has a copy of $Prof(u)$ in its memory, $timestamp$ contains the time of the election of $PoS(u)$, $IsOnline$ is a boolean flag that indicates whether $PoS(u)$ is online or not and $IsOutdated$ indicates whether $PoS(u)$'s version of $Prof(u)$ is outdated or not, which is whether the last change to $PoS(u)$ was made before $PoS(u)$'s disconnection from the system or not.

When a node is elected as PoS for some user u he adds an entry in user u 's entry list in the PoS Table. When he destroys its local copy of a user

profile it deletes the correspondent entry in the PoS Table. Through the PoS Table each user always knows the current PoS of his friends and can access the newest version of their profiles currently online. We assume there is an authentication mechanism that allows the access to the user u 's PoS information only to his friends.

The association between a user and his PoS is stored in a DHT [13]: friends of an offline user u can always access the DHT and get the IP address of $PoS(u)$ to retrieve $Prof(u)$.

Friends of u might exchange his data, but if they are not mutual friends themselves they cannot host each other's data. We assume user u 's knowledge of the social network corresponds to his Dunbar-based ego network, which is the set $df(u)$, the state (online/offline) of each user $\in df(u)$, and the tie strength of each link in $\{(x, y) \mid x, y \in \{u\} \cup df(u)\}$. A user u also knows the set of nodes n in $PoS^{-1}(u) = \{n \mid u \in PoS(n)\}$. When u disconnects from the system, it exploits $PoS^{-1}(u)$ to elect a new PoS for each node n in $PoS^{-1}(u)$ that is offline at that moment. To implement the election of a new PoS for n , u must also know the list of *all* friends of n and all information about them required to implement the PoS selection strategies that we will describe in Section 4, for instance the tie strength and the average session length of these nodes.

Another challenge of our approach is the robustness against single-node failure. A PoS may voluntary disconnect suddenly or because of different types of failure without being able to pass a replica of its data to another node. To address this goal we keep two online copies of each user's profile at any time: an online node n must have one online PoS so that a copy is still present in the system also in case of failure of n or of $PoS(n)$. For the same reason an offline node with more than one online friend must have two online PoS. Since dealing with the problem of consistency in detail is beyond the scope of the paper we assume the consistency of copies of the same profile on different PoS is always guaranteed by some external mechanism. We further assume there is a recovery mechanism for detecting faults and crashed nodes and elect new PoS to replace them.

4 PoS Selection Strategies

The choice of friends where to store replicas of users' profiles is a fundamental issue in our system. We consider social and structural properties of the social graph, and the user's behaviour to define our approach. We assign to each node a score and we propose different strategies which can be adopted to achieve different user's performance objectives. We define a numerical value, called *SocialScore*, measuring the users' suitability as PoS of a specific node. The $SocialScore(SS)$ of a user x which is a Dunbar neighbour of a user n is defined as follows:

$$SS_{ego_n}(x) = \alpha \cdot TieStrength_{nx} + \beta \cdot CG_n(x) + \gamma \cdot MSL(x)$$

where all the terms $TieStrength_{nx}$, $CG_n(x)$ and $MSL(x)$ are normalized and α , β and γ are weights autonomously chosen by the users according to the relevance given to each criteria.

The *ConnectionGain* ($CG_n(x)$) for an online node x in the ego network of a user n is the gain in terms of trusted connections obtained by the election of x as $PoS(n)$ and is proportional to the number of common neighbours of x and n . If $CG_n(x)$ has an high value, then x may transfer the profile of n to neighbours of n through a trusted connection.

$MSL(x)$ is the *Medium Session Length* and represents the average duration of a user session. We approximate this measure so that it is able to distinguish at least between users almost always connected to the social network (e.g. through mobile devices), those that connect only for short periods of time and those that use the social network as a means of social interaction and therefore have quite long sessions on average. We will name *Fair Strategy* the strategy that gives to all the weights the value 1.

4.1 Maximizing the Trust (MaxTrust)

In this approach the social score maximizes the trustness (β and γ are equals to 0). A user selects as his PoS those friends with whom he has the strongest relationships, in order to store data on the most trusted friends. Hence, the online friend with the highest value of tie strength with the user is selected as his new PoS. The value of the tie strength between the nodes may be computed as a combination of many factors such as the contact frequency between the two users, the number of likes, posts, comments, private messages, tags, etc. We exploit the approach of [14] to compute the tie strength.

4.2 Maximizing the Trusted Connections (MaxTrustConnect)

In this strategy the social score is obtained with α and γ equals to 0 and β equals to 1. We select as PoS for a user u the node y in $df(u)$ with the highest value of the *ConnectionGain*. The rationale for this strategy is that users who are interested in accessing the profile of a user u are his friends and the elected PoS should send the profile of u to these nodes, when they require it. If these nodes are, in turn, friends of the PoS, the number of connections established between nodes that are not Dunbar ego friends each other is reduced. We call these kind of connections as *untrusted*, whereas we call *trusted* all the connections established between a user u and one of his Dunbar ego friends. The set of online nodes whose connection to y to access $Prof(n)$ would be trusted is the *Trusted Connection Set*, which is defined as $TCS(y, n) = \{y\} \cup \{x \mid x \in df(y) \cap df(n) \wedge x.isOnline()\}$. Since we elect two PoS for an offline user u , only online nodes in TCS that haven't already got a trusted connection to the other PoS of n contribute to the *ConnectionGain*. So we define the *Uncovered Trusted Connection Set (UTCS)*, as $UTCS(y, n) = TCS(y, n) - \{x \mid PoS^*(n) \in df(x)\}$ where $PoS^*(n)$ is the *online* PoS of the user n . We select as Pos for a user u the node y in $df(u)$ with the highest value of the connection gain which is formally defined as:

$$CG_n(y) = \sum_{j \in UTCS(y,n)} f_{jn}$$

In order to maximize the number of trusted connections between ego friends, we approximate the possibility that a node n contacts $PoS(u)$ with its contact frequency f_{nu} . Note that the probability that a user belonging to $ef(u) - df(u)$ contacts $PoS(u)$ is negligible.

Consider a node $p \in df(u)$ that has been selected as one of u 's PoS. Common ego friends between u and p can access $Prof(u)$ through their trusted link to p .

Example 1. The values of CG_n calculated for the nodes of the graph in Fig. 1(a) when no $PoS(X)$ is still elected are shown in Fig. 1(b). In this scenario node D will be elected as $PoS(X)$. The values of CG_n calculated after D's election as $PoS(X)$ are shown in Fig. 1(c). Node I would be elected as the second PoS for node X at the moment of its disconnection from the system. In the example we would obtain $PoS(X) = \{D, I\}$ and 8 out of a total of 11 nodes in $EN(X)$ would be able to access $Prof(X)$ through a trusted connection. \diamond

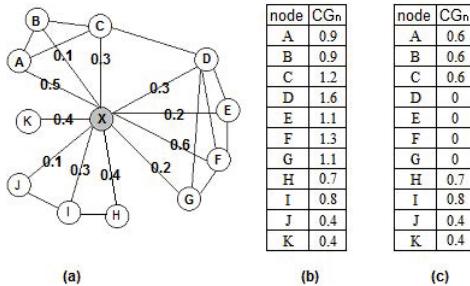


Fig. 1. (a) X's ego network $EN(X)$, (b) nodes' CG_X when no $PoS(X)$ is still elected, (c) nodes' CG_X after D's election as $PoS(X)$

4.3 Minimizing the Number of Profile Transfers (MinTransf)

This strategy minimizes the number of times a profile has to be transferred to another node (α and β equal to 0). We consider the user's behaviour as basic property of this strategy, in particular the session length. Differently from [3] we don't assume that users' online time is known *a priori*, but we need an estimation of the time an online user will remain online before his disconnection. We define a greedy strategy choosing as PoS of a node its Dunbar neighbour which is online and that is more likely going to remain connected to the system for the longest time interval.

Algorithm 1. Best PoS Selection

```

1: function SELECTBESTPOS( $p$ ,  $Set$ )
2:   if ( $\exists x \in Set \mid x.isOnline()$ ) then
3:     get  $SS_{ego_p}(v) \forall v \in Set \mid v.isOnline()$ ;
4:     select  $n \mid SS_{ego_p}(n) = \max\{SS_{ego_p}\}$ ;
5:     return  $n$ ;
6:   else
7:     return null;
8:   end if
9: end function

```

Algorithm 2. PoS election for a node p

```

1: function ELECTION( $p$ )
2:    $Set = df(p) - PoS(p)$ ;
3:    $n = SelectBestPoS(p, Set)$ ;
4:   if  $n \neq null$  then
5:     send  $<Prof(p), ef(p), TieStrength_{p*}>$  to  $n$ ;
6:     update  $PoSTable(p)$ ;
7:   else
8:      $Set = ef(p) - PoS(p)$ ;
9:      $n = SelectBestPoS(p, Set)$ ;
10:    if  $n \neq null$  then
11:      send  $<Prof(p), ef(p), TieStrength_{p*}>$  to  $n$ 
12:      update  $PoSTable(p)$ ;
13:    end if
14:  end if
15: end function

```

5 PoS Election Algorithms

In this section we describe the algorithms for the election of the PoS which must guarantee that each online node has one PoS among its online friends and that each offline node with more than one online friend has two PoS among them. First, we define an auxiliary procedure **SelectBestPos()** that, given user p and a given set of nodes Set , selects the best PoS for p among them.

Algorithm 1 gets $SS_{ego_p}(v)$ for all online nodes in the set and elects as the new $PoS(p)$ the node with the highest Social Score. If all nodes in the set are offline the algorithm returns null, otherwise the selected node is returned.

Algorithm 2 shows the procedure to elect a new PoS for a node p . The best PoS is chosen among $df(p)$ at first and among $ef(p)$ if all nodes in $df(p)$ are offline. Selected PoS receives from p $Prof(p)$, $df(p)$ and all the tie strength values between p and one of its friends ($TieStrength_{p*}$). Afterwards, p 's entry in the PoS Table is updated with the information about p 's new PoS. Algorithm 2 is executed when an online node p receives a disconnection notification from a node that is its only current $PoS(p)$ and when a node is going to disconnect from the system and needs to elect its second PoS.

Let us now consider the procedure executed by a node n which is going to disconnect from the system and which is the PoS for at least a user. n notifies its disconnection to online nodes that belong to $Pos^{-1}(n)$ and these nodes will be able to elect a new PoS on their own. As far as concerns offline nodes $\in Pos^{-1}(n)$, n must select a new PoS for each of them among their neighbours. Notice that n received the list $ef(p)$ at the moment of its election as $PoS(p)$. As for the election

Algorithm 3. PoS election for an offline node executed by a disconnecting node

```

1: function ONDISCONNECT( $p$ )
2:   for all  $u \in Pos^{-1}(p) \mid !(u.isOnline())$  do
3:      $Election(u);$ 
4:   end for
5: end function

```

procedure the best PoS is the node with the highest Social Score. Algorithm 3 shows the procedure previously described.

Example 2. Consider the graph in Fig. 2 where all nodes except F are online: cylinders next to a node represent data stored by it. Each online node has a PoS in the network and F, which is offline, has its profile replicated on two different nodes. Suppose node A is going to disconnect from the system. Before its disconnection it controls all nodes that belong to $Pos^{-1}(A)$ and detects that C and H are online and will be able to elect a new PoS by themselves. But since F is offline A must elect a new PoS for F, which will be chosen among F's Dunbar ego friends, namely {E, G, H, I}. \diamond

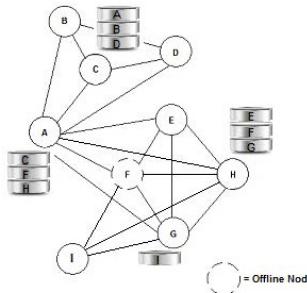


Fig. 2. PoS re-election for an offline node

6 Simulation Methodology

The lack of a dataset containing both temporal information about users' online sessions and information regarding the structure of the social network and the interactions between the users is the major limitation for the study of data availability in DOSNs. For this purpose, we have developed a crawler to log the sessions of Facebook's users and we have integrated it with data extracted from the dataset published by Zhao et al.¹.

¹ This dataset is publicly available for research at

<http://current.cs.ucsb.edu/facebook/> referred as "Anonymous regional network A".

6.1 The Facebook'14 Data Set

We have implemented a Facebook application (available at www.socialcircles.eu) which allows the user to display its ego network structure, the geographical location of his contacts and to compute statistics about the interactions with his friends. The application collects information about:

- *network topology and users' profiles*: the profile of each registered user is logged together with the list of his friends.
- *users' interactions*: the application logs posts, photos, comments, likes and tags which have been posted within six months before the crawling end time (May 2014).
- *availability traces*: these are collected by analysing the state of the user, i.e. *Online*, *Offline*, or *Idle* if the user has not generated any event at least during the last 10 minutes. This information is collected by monitoring the only temporal information provided by Facebook, which is the activity of the users in the chat.

In order to avoid an excessive number of Facebook API calls, we decided to collect information about a subset of 25 users (called *egos*) selected among the registered ones (approximately 300 users). In order to maximize the number of data collected from the users we selected the 10 users with the highest number of friends and 15 further randomly chosen users. We monitored the selected users for 6 days, from April 30 to May 6, 2014, and we registered the availability state of each user and of his friends with a 5-minute frequency. In order to get a one-minute-step availability trace for each user, we considered a 5-minute temporal window in which the user's state does not change, centred on the time of data collection.

The contact frequency between each couple of friends was computed by dividing the number of interactions between an ego and the alter by the duration of their social relationship. Since Facebook does not allow us to acquire information related to the start time of a social relationship, we estimated the duration of the social relationship as the time elapsed between the first interaction between the users and the end time of crawling (in hours). Since no timestamp is paired with some interactions (such as Tags and Likes), we approximated the correspondent friendships' duration with the overall duration of the period considered by the crawling process (six months). Then, we exploited contact frequencies to calculate the tie strength from a user j to a user k , as proposed in [14]

We define as Dunbar ego network of a user the set of his friends for whom the value of contact frequency is greater than 0, limited to at most 150 contacts.

Finally, only the tie strength between an ego and his alters and viceversa have been computed, since information about interactions between alters was not available.

Network Statistics. We have collected 3,203 relationships and 29,616 interactions, and the resulting graph contains 13,693 users, along with 299,559 friendship relations. The number of friends of the 25 egos in our sample ranges between

76 and 1,321 (Mean = 558.84, Standard Deviation = 340.32). Each ego interacts only with a small subset of his friends: we call *active friends* the subset of friends with whom the ego has had at least one interaction during the 6-month period of analysis.

Figure 3(a) depicts the number of users in an idle status, the number of online users and the total number of users connected to the system during the observed period. The number of users connected simultaneously does not exceed 2,645. Drops due to the diurnal pattern are clearly visible in the graph: this can be explained by the fact that the vast majority of the users belongs to the same geographical area.

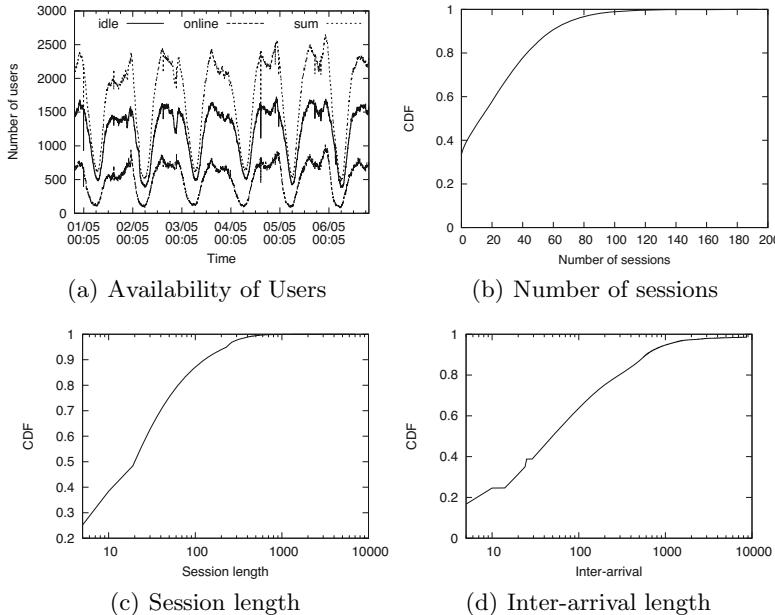


Fig. 3. Information about users' sessions

Session Statistics. We studied availability traces to analyse how frequently, for how long and how many times the sampled users and their friends connect to the system.

The 34% of the users did not use the OSN during the period of crawling. The 50% of the users had less than 13 different sessions during the observed period (an average of 2 sessions per day). Fig. 3(b) shows the Cumulative Distribution Functions (CDF) of the number of sessions per user. Fig. 3(c) and Fig. 3(d) show the CDF of the *session length* and of the *inter-arrival session times* respectively. The session duration plot shows a large variation in the OSN usage among users. The 38% of the users' session are shorter than 10 minutes while the vast majority

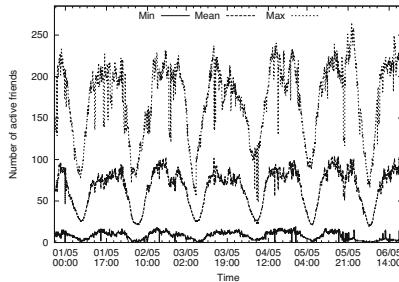


Fig. 4. Number of active friends for ego nodes

of the users (90%) spend online no more than 125 minutes at a time. The inter-arrival times illustrates the interval between the moment a user departs from the system and his next arrival. Figure 3(d) shows that 50% of users have an inter-arrival times shorter than 1 hour. These observations are consistent with measures previously made in [11] and [15].

Fig. 4 shows the minimum, average and maximum number of online active friends of the 25 users during the observed period. The number of online active friends shows a strong daily pattern similar to the one in Figure 3(a), where the number of users simultaneously online in the same network never exceeds 265. The average number of online active friends is 65.74 and it remains stable over the entire observation period. Since the minimum number of online active friends is 0, there are some short periods of time in which some of the users have all their active friends offline.

6.2 The Zhao's Dataset

We were able to compute the tie strength between each ego and its alters, while it was not possible to compute the value of tie strength between a pair of alters. Furthermore, only the ego networks of the 25 users we considered are complete, while the ego networks of the alters of these users include only the common friends between each alter and an ego. For this reasons, we integrated the information of the Facebook'14 dataset with the *Zhao's Dataset*. We extracted a strongly-connected component of the same size of the network in *Facebook'14 Dataset* (13,693 users) by a degree-based progressive sampling technique and paired each node with an availability trace chosen uniformly at random from a user of the *Facebook'14 Dataset*. From the interaction graphs we extracted the frequencies of contact between each pair of users, normalized them, and computed the correspondent tie strength according to [14]. Nodes of the extracted component are generally characterized by an high degree. We introduced a threshold to the tie strength value associated to the relationship from node a to node b . If the value is below the threshold, b is not inserted in a 's Dunbar ego network, while it is inserted in the ego network of a .

7 Simulation Results

We built a Java-based simulation for our system using the PeerSim P2P simulator²: we created a 13,693-node network with a 1-to-1 mapping between the nodes and the users of our dataset. Each node has the same ego network of the correspondent real user and is associated to his availability trace. The simulation replicates the actual online status of the users by considering the time stamp information.

7.1 Performance Metrics

We exploit several performance metrics to evaluate our system:

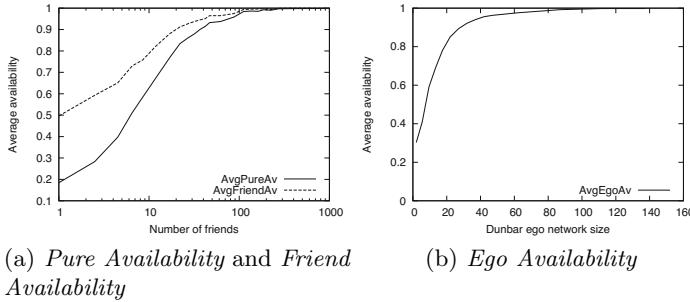
1. *Pure Availability*: [3] fraction of time in a day a user's profile is reachable through his PoS, that is the union of times the user has at least one online PoS.
2. *Friend Availability*: [3] fraction of time in a day a user's profile is reachable through his PoS from his friends. In other words, since the profile of a user is accessible only from his friends, this metrics focus on the fraction of time the profile of a user is available when his friend are online. In a social network higher *Friend Availability* (even with a lower *Pure Availability*) is desirable.
3. *Ego Availability*: fraction of time in a day a user's profile is reachable through one of his PoS belonging to his Dunbar neighbours.
4. *Average Load*: [3] average number of profiles stored on the user when he is online. A good election strategy should balance the load on different nodes.
5. *Average PoS Tie Strength*: average tie strength value between the user and his online PoS. A high value means that the PoS have been chosen among more trusted friends.
6. *Average Number of Elections*: average number of PoS elections made by a user.

7.2 Results

Pure Availability and Friend Availability. *Pure Availability* and *Friend Availability* in the *Facebook Dataset* are shown in Figure 5(a). Since availability depends on users' traces and only marginally on PoS selection strategies, results are very similar for all the proposed strategies. Hence, we show only the graphs corresponding to the *MaxTrustConnect* selection strategy.

Pure Availability increases monotonically and an availability of 90% is achieved for nodes with more than 40 friends. As we expected, *Friend Availability* is always greater than or equal to *Pure Availability*. It is remarkable to note that also a user with a relatively small number of friends (e.g. 20) achieves more than 90% of *Friend Availability* on average. For nodes with a very small number of friends (i.e. < 10) the difference between *Pure Availability* and *Friend Availability* is

² <http://peersim.sourceforge.net/>

**Fig. 5.** Statistics about availability

remarkable, since neither the node nor its friends are online for long periods of time.

Ego Availability has been evaluated only on the *Zhao's Dataset*, because it requires the value of the tie strength between each pair of users. Fig. 5(b) shows the average *Ego Availability* as a function of the ego network size. *Ego Availability* grows monotonically and achieves 90% for nodes with at least 30 ego friends: this means that also nodes with only 30 friends in their Dunbar based ego network almost always elect as PoS nodes belonging to their Dunbar ego networks.

Average Load. Since a user's PoS are always chosen between his online friends, we expect load increases proportionally to users' ego network size and to the percentage of time they spend online in the system.

Fig. 6(a) and Fig. 6(b) show, respectively, plots of the *Average Load* of the *Facebook'14* and of the *Zhao's Dataset* as a function of the percentage of time a user has spent online. In both graphs we can notice that users that spend more time online generally have a higher load with respect to those that are connected less frequently. Since *MinTransf* selects PoS with high *Medium Session Length*, it seems to be reasonable that nodes that are often offline are chosen as PoS less frequently than with other strategies.

The maximum load is about 50 profiles for the *MinTransf* strategy in *Facebook'14 Dataset* and about 30 profiles for the same strategy in the *Zhao's Dataset*. Since both datasets use the same availability traces, results may appear surprising. Instead, they can be explained by considering the different degree distribution of the two datasets: in *Facebook'14* there are a few nodes with a very high degree which may be chosen by many nodes as their PoS. On the other hand, degrees in *Zhao's Dataset* are more uniformly distributed. Fig. 6(c) shows the plot of the average load in the *Zhao's Dataset* as a function of the Dunbar ego network size and by considering the different strategies presented in Section 4: the maximum load achieved for complete ego networks is 24 profiles with the *MaxTrustConnect* strategy. Since the *MaxTrustConnect* strategy selects PoS according to their *Connection Gain*, which is directly proportional to the ego network size, this result matches our expectations. Finally it is remarkable to note that nodes with less than 80 ego friends have an average of less than 10 profiles stored in their memory with all the strategies.

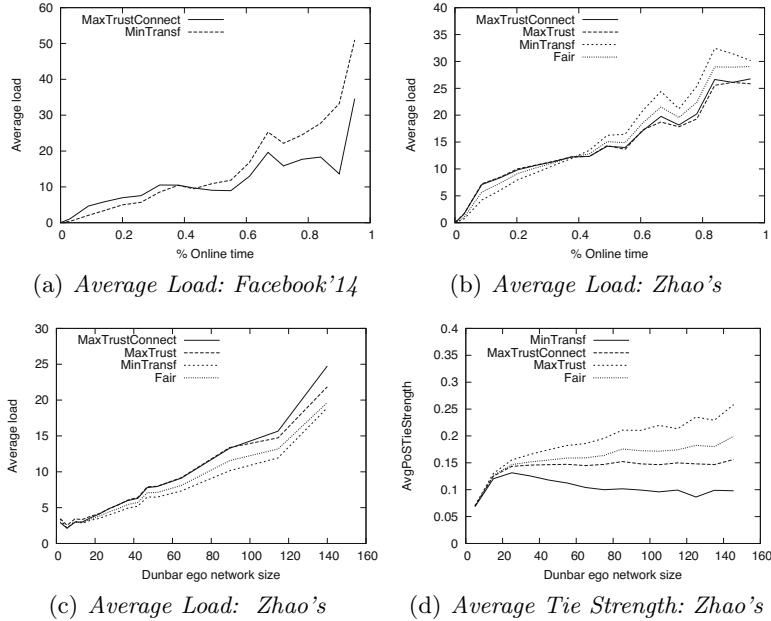


Fig. 6.

Average PoS Tie Strength. Fig. 6(d) shows a plot of the average tie strength values between the nodes in the *Zhao's Dataset* and their online PoS as function of the Dunbar ego network size. The average value of the tie strength returned by the different strategies is almost the same for small sized Dunbar ego networks, since the choice of PoS is always restricted among nodes that are online at the moment of the election. As the Dunbar ego network size grows, the difference between the average tie strength values obtained with the different strategies increases and, as expected, the highest tie strength value is obtained with the *MaxTrustConnect* strategy. The histogram in Fig. 7(a) shows the average value of tie strength between the 25 egos in *Facebook'14* and their online PoS. The average values of the different strategies are compared with the average tie strength between the same 25 egos and all their ego friends (the light blue bar on the right). As before, the average PoS tie strength is much higher for *MaxTrustConnect* than for the other strategies.

Average Number of Elections. The average number of elections made by each user is shown in the histogram of Fig. 7(a). All the strategies are compared for the *Zhao's Dataset*. The minimum average number of elections is obtained with the *MinTransf* strategy that exploits information about users' sessions for PoS elections. Currently, medium session length is computed during the simulation, by considering previous sessions of the users during the considered period. We believe that the number of elections may be further reduced, by using longer periods of times and/or by refining the methodology used to predict users' availability,

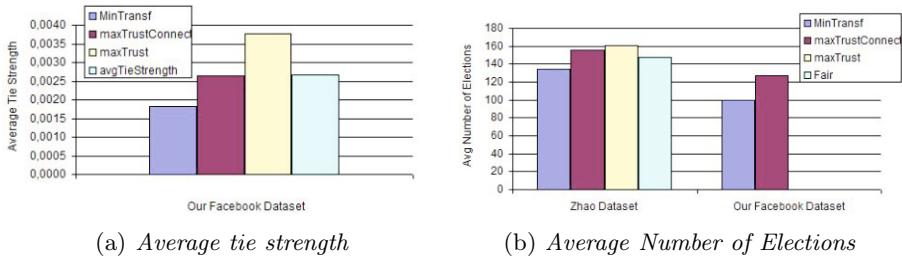


Fig. 7.

8 Related Work

Recently, several P2P architectures for DOSNs have been proposed. LifeSocial [16] is a P2P-hosted OSN where users employ public-private key pairs to encrypt profile data that are securely stored in a FreePastry-based DHT. PeerSon [17] is a distributed infrastructure for social networks whose focus is related to security and privacy concerns. It exploits encryption, direct data exchange and access control. Safebook [18] addresses privacy in OSNs by storing encrypted profile content in a P2P storage infrastructure.

Most of DOSNs rely on servers or other external services to guarantee data availability. Diaspora [19] is a social network in which users' profiles are hosted on servers that are administrated by individual users who decide themselves where their information will be stored. SuperNova [20] is a DOSN that manages the availability by using a super-peer architecture that provides highly available storage.

Others DOSNs built on Friend-to-Friend (F2F) networks do not require any complex encryption mechanism for data management. In [21] the authors argued that a user should choose the nodes with whom it shares data based on existing social links instead of choosing them at random.

Important research efforts have been focused on studying the problem of data availability in DOSNs. Friendstore [17] is a social back-up system that ensures availability by storing data on trusted nodes only. [3] proposes a replication strategy in a DOSN in which replicas of users' profiles are stored only on a set of trusted proxies. [3] and [4] propose different replica-placement techniques to guarantee data availability. Instead of replicas in [22] authors propose to store data blocks on nodes by considering their availability. [23] and [24] present new techniques to predict users' availability based on their historical behaviour and authors in [15] demonstrate that online presence of OSN users tend to be correlated to those of their friends. Only a few studies have been made on data availability in F2F storage systems: [25] shows that F2F systems cannot guarantee high data availability for users with a few friends online and [5] focuses on the impact of availability correlation and very small friend sets on F2F systems' data availability.

9 Conclusion

This paper has presented a distributed storage support which guarantees the users' data persistence of a distributed online social network. The main goal of our system is to guarantee that users' data are stored on trusted friends. Each node dynamically elects a minimal set of Point of Storages among his friends. User data are dynamically transferred between online users in order to maximise the availability of users' profiles in the social network. We plan to extend our system in several directions. We will define a proper set of strategies to guarantee the consistency of the copies of the users' data. Furthermore, while in our experiments we have used as first approximation for the future availability of a user its medium session length, we plan to refine this measure by considering further statistical metrics.

References

1. Datta, A., Buchegger, S., Vu, L.-H., Strufe, T., Rzadca, K.: Decentralized Online Social Networks. In: *Handbook of Social Network Technologies and Applications*, pp. 349–378 (2010)
2. Adya, A., Bolosky, W.J., Castro, M., Cermak, G., Chaiken, R., Douceur, J.R., Howell, J., Lorch, J.R., Theimer, M., Wattenhofer, R.P.: Farsite: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. In: Proc. of the 5th Symp. on Operating Systems Design and Implementation, OSDI, NY, USA, pp. 1–14 (2002)
3. Narendula, R., Papaioannou, T., Aberer, K.: A Decentralized Online Social Network with Efficient User-Driven Replication. In: *International Conference on Social Computing (SocialCom)*, pp. 166–175 (2012)
4. Schiöberg, D., Schneider, F., Trédan, G., Uhlig, S., Feldmann, A.: Revisiting Content Availability in Distributed Online Social Networks. CoRR, vol. abs/1210.1394 (2012)
5. Gracia-Tinedo, R., Sanchez Artigas, M., Garda Lopez, P.: Analysis of data availability in F2F storage systems: When correlations matter. In: IEEE 12th International Conference on Peer-to-Peer Computing, pp. 225–236 (2012)
6. Everett, M., Borgatti, S.P.: Ego network betweenness. *Social Networks* 27(1), 31–38 (2005)
7. Dunbar, R.I.M.: The social brain hypothesis. *Evolutionary Anthropology: Issues, News, and Reviews* 6(5), 178–190 (1998)
8. Roberts, S.G.B., Dunbar, R.I.M., Pollet, T.V., Kuppens, T.: Exploring variation in active network size: Constraints and ego characteristics. *Social Networks* 31(2), 138–146 (2009)
9. Arnaboldi, V., Guazzini, A., Passarella, A.: Egocentric Online Social Networks: Analysis of Key Features and Prediction of Tie Strength in Facebook. *Computer Communications* 36(10-11), 1130–1144 (2013)
10. Sutcliffe, A., Dunbar, R., Binder, J., Arrow, H.: Relationships and the social brain: Integrating psychological and evolutionary perspectives. *British Journal of Psychology* 103(2), 149–168 (2012)
11. Benevenuto, F., Rodrigues, T., Cha, M., Almeida, V.: Characterizing User Behavior in Online Social Networks. In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, New York, NY, USA, pp. 49–62 (2009)

12. Mega, G., Montresor, A., Picco, G.: Efficient Dissemination in Decentralized Social Networks. In: IEEE International Conference on Peer-to-Peer Computing (P2P), pp. 338–347 (2011)
13. Balakrishnan, H., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Looking up data in p2p systems. *Commun. ACM* 46(2), 43–48 (2003)
14. La Gala, M., Arnaboldi, V., Conti, M., Passarella, A.: Ego-net digger: a new way to study ego networks in online social networks. In: Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research, HotSocial 2012, New York, NY, USA, pp. 9–16 (2012)
15. Boutet, A., Kermarrec, A.-M., Le Merrer, E., Van Kempen, A.: On the Impact of Users Availability in OSNs. In: Proceedings of the Fifth Workshop on Social Network Systems, SNS 2012, New York, NY, USA, pp. 4:1–4:6 (2012)
16. Graffi, K., Gross, C., Stingl, D., Hartung, D., Kovacevic, A., Steinmetz, R.: Life-Social.KOM: a secure and P2P-based solution for online social networks. In: IEEE CCNC, pp. 554–558 (2011)
17. Tran, D.N., Chiang, F., Li, J.: Friendstore: cooperative online backup using trusted nodes. In: SocialNets 2008: Proceedings of the 1st Workshop on Social Network Systems, pp. 37–42. ACM, New York (2008)
18. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: a Privacy Preserving Online Social Network Leveraging on Real-Life Trust. *IEEE Communication Magazine* 47(12) (2009)
19. <https://diasporafoundation.org/>
20. Sharma, R., Datta, A.: SuperNova: Super-peers Based Architecture for Decentralized Online Social Networks. *CoRR*, vol. abs/1105.0074 (2011)
21. Frank, J.L.: F2F: reliable storage in open networks (2006)
22. Pamies-Juarez, L., García-López, P., Sanchez-Artigas, M.: Heterogeneity-aware erasure codes for peer-to-peer storage systems. In: International Conference on Parallel Processing, ICPP, pp. 412–419 (September 2009)
23. Le Blond, S., Le Fessant, F., Le Merrer, E.: Finding Good Partners in Availability-Aware P2P Networks. In: Guerraoui, R., Petit, F. (eds.) *SSS 2009. LNCS*, vol. 5873, pp. 472–484. Springer, Heidelberg (2009)
24. Mickens, J.W., Noble, B.D.: Exploiting Availability Prediction in Distributed Systems. In: Proceedings of the 3rd Conference on Networked Systems Design & Implementation, NSDI 2006, Berkeley, CA, USA, vol. 3, p. 6 (2006)
25. Sharma, R., Datta, A., Dell'Amico, M., Michiardi, P.: An empirical study of availability in friend-to-friend storage systems. In: IEEE International Conference on Peer-to-Peer Computing (P2P), pp. 348–351 (August 2011)

A Cooperative Information System for Managing Traffic Incidents with the PAUSETA Protocol

Miguel Prades-Farrón, Luis A. García, and Vicente R. Tomás

Engineering and Computer Science Department,
University Jaume I, Castellón, Spain
{farron,garcial,vtomas}@uji.es
<http://www.aia.uji.es>

Abstract. When a serious traffic incident happens several independent agencies must be coordinated in order to allocate the necessary resources to attend it. Currently, the coordination among these agencies is done manually, increasing the response time. In this paper is presented an information system suitable to be applied to reach agreements in a distributed and autonomous way among agencies about which an agency should give which resource to manage a traffic incident. This information system is guided by the PAUSETA protocol, which executes a distributed combinatorial auction for resource allocation. By using this protocol several issues around private information about resource features and availability and administrative and legal competences are individually managed by each agency. This information system has been implemented following a multi-agent software architecture. The prototype has been tested for a real traffic incident scenario in Castellón (Spain). The results obtained are consistent with respect to the ones it might provide an optimum centralized system. However this optimum centralized system is impossible to apply due to the inherent distributed nature of the problem addressed.

Keywords: Cooperative Information Systems, Traffic Management Plans, Combinatorial Distributed Auctions.

1 Introduction

Serious traffic incidents are tragic situations that cause dead and wounded people every year. If the traffic incident is not attended, the probability of avoiding the death of a seriously injured person decreases with the time. This time is known as the golden hour [1]. In addition, the traffic in the area where the incident occurs is congested more and more over time after the incident. To stabilize the injured people and reduce traffic congestion, a set of resources must be allocated. The response time between occurring the traffic incident and required resources that are assigned is crucial. To address the incident, the public administration uses Traffic Management Plans (TMP) [2][3][4][5]. A TMP describes the scenarios, measures and actions to be developed to treat a traffic incident. In the scenario part, features and location of the traffic incident are described. The measures

describe the set of procedures that must be applied to deal with the described scenario. Finally, the actions describe the set of operations and tasks that each involved agency has to deploy to execute the selected procedures from the TMP (see Fig. 1).

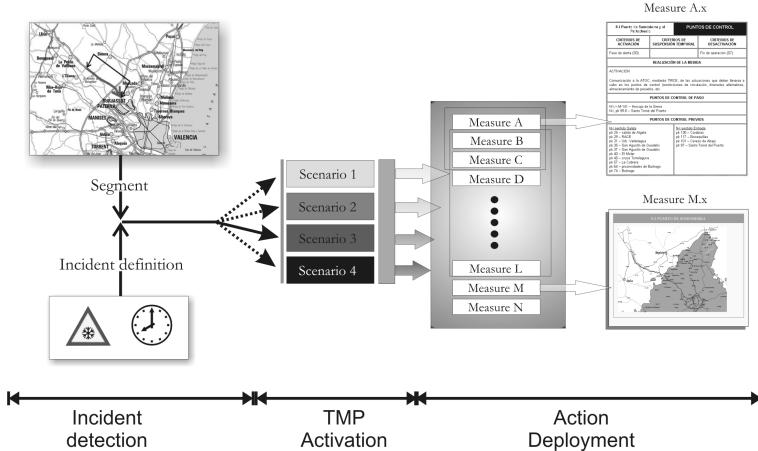


Fig. 1. In this figure the phases of a TMP activation are presented. First, once the responsible agency of the road management has detected and validated an incident, it activates the TMP for the current location, and by using this incident information, it determines the current scenario. Then, all agencies involved in the TMP start the coordination to develop the measures. To develop a measure, agencies must deploy individual actions. A measure only can be deployed when all involved agencies are ready to activate their individual actions.

However, agencies that provide resources for the action and measurement parts are independent, for example: hospitals, fire stations, traffic police headquarters, fire stations, civil works, etc. These agencies should coordinate with each other to decide who has to provide each resource. The usual way to coordinate these agencies is by means of traffic operators of a centralized public administration which manually negotiates with them. This way of negotiations is very time consuming which exacerbates the response time. Moreover, these involved agencies may have different competences (nation-wide, region-wide or local) to deploy the actions of a TMP and these competences cannot be transferred with each other. Therefore, a tool to automate the negotiation process and also the one which takes into account the interests and competencies of each independent agency would accelerate this coordination and therefore, the response time.

A widely used method to negotiate the allocation of resources is through combinatorial auctions [6]. An auction is a mechanism to assign a resource to the most interested bidder in getting it among a set of suitable bidders. Combinatorial auctions are a type of auction in which the auctioneer offers several resources

simultaneously and the bidders can bid on packages of these resources instead of individual resources. A package is a set of resources that bidders are interested in. Bidding over packages, bidders could obtain more profits because they can exploit the synergies that may exist between the resources in a package, i.e. positive value of the synergy expresses that the value of a package is major than the sum of its resource values. There are several application domains in which combinatorial auctions are used to allocate resources as: bandwidth [7], energy slots [8], time slots [9], truckload transportation [10], coordination of team mobile robots[11], among others.

Combinatorial auctions can be classified into centralized and distributed auctions. In a centralized combinatorial auction, the auctioneer has the role of determining which combination of resources, from the received bids, is the winner. It is a well-known NP-Complete [12] problem. In a distributed combinatorial auction, the winner determination problem is calculated between all bidders and therefore, the role of the auctioneer might be suppressed. Distributed auctions have several advantages with respect to the centralized auctions: 1) a bidder does not have to trust in the combination of resources made by the auctioneer, since the auctioneer can manipulate it, 2) it is more robust against the loss of communications than centralized auctions, since these centralized auctions can not give a solution if the communication with the auctioneer is lost and 3) bidders can keep certain privacy data, as they only share the necessary information with others. A protocol for distributed combinatorial auctions is the PAUSETA protocol (Progressive Adaptive User Selection Environment by Type Agreements) [13]. This protocol is a extension of the PAUSE [14] protocol and defines the rules for a distributed and multi-unit combinatorial auction, i.e. it allows to auction for multi-unit resources. A multi-unit resource is a set of resources of the same type. This peculiarity allows to apply the PAUSETA protocol in the management of resources for traffic incidents, since the needed resources to apply a TMP are given in this way. For example, to attend the injured people in a traffic incident, two ambulances are needed, i.e. two units of the ambulance resource type.

In this paper, a cooperative information system [15] that aims to negotiate the allocation of the needed resources to treat a traffic incident has been designed. Furthermore, an ontology has been designed so that the individual involved agencies in the negotiation can be coordinated. The negotiation mechanism used is the PAUSETA protocol. Finally, the designed cooperative information system has been implemented by a multi-agent architecture to be evaluated.

This paper is organized as follows. In the next section, a cooperative information system for automatic traffic incident management is described. Specifically, it describes the PAUSETA protocol and the components of the traffic incident ontology. In the Sect. 3, a multi-agent architecture that implements both the PAUSETA protocol and the mentioned ontology is exposed. Next, a real environment around Casellón harbour has been modelled. In Sect. 5, a series of tests have been carried out on this environment in order to evaluate the performance of the multi-agent system. Finally, the conclusions and future work are exposed.

2 A Cooperative Information System for Automatic Traffic Incident Management

In this section, the elements of a cooperative information system to address traffic incidents are described. This system is composed by a negotiation protocol and an ontology for allocation resources in traffic incidents.

2.1 The PAUSETA Protocol

PAUSETA is a protocol for distributed combinatorial auctions that runs in sequential stages and each stage involves the execution of several rounds. PAUSETA is designed to deal with t sets of multi-unit resources instead of managing only individual resources. Let r_i , $1 \leq i \leq t$, denotes the multi-unit resource i . Each r_i represents a set of concrete resources of the same type. For example, r_1 can represent the type of resource *traffic patrols* and there can be three concrete available traffic patrols: p_1, p_2, p_3 . Each bidder values each multi-unit resource according to their private interests. For example, the closer the resources are to where it should be assigned, the stronger the interest of the agency. However, not all the resources of the same type have the same assessment. It depends on the amount of resources of the same type the bidder has available. As a bidder runs out of resources of the same type it will be more reluctant to allocate them, as these resources may be more useful for future emergencies. In a PAUSETA auction, each bidder tries to maximize the current best bid by making bids with the bigger values of the resources that are still available. The number of stages in PAUSETA is t , as many as different types of resources are required, i.e. multi-unit resources. Beginning in the second stage, each bidder must make a Complete Composite Bid (CCB). A CCB is a set of bids made by one or more bidders covering all m required resources. The value of the CCB is the sum of bids that compose it. In each stage k , $2 \leq k \leq t$, it is possible for a bidder to make bids for packages of size up to k types of resources, i.e., in stage 1, a bidder can make bids for just one type of resource, in stage 2 a bidder can make bids for packages of one or two types of resources, and so on. The figure of the auctioneer does not exist in the PAUSETA protocol. So, each bidder must ensure that the rest of bidders follow the rules of the auction. Therefore, the bids made by a bidder are broadcasted to the other ones. A detailed explanation of the PAUSETA stages is exposed below.

The First Stage. Figure 2 shows the exchanged messages between the agents in the first stage of the PAUSETA protocol. The first stage is different from all others. At this stage each, bidder tries to provide all necessary resources for itself, i.e. if the bidder has enough resources, it must bid for all the required resources; on the contrary, if it does not have enough resources, it must bid for the maximum number of resources that it can provide. In a single round, all bidders submit their bids to the rest of them. The bids received by each bidder are stored in a local set called *the Set of All Bids (SAB)*. These stored bids will be used by the bidders in the successive steps to make CCBs.

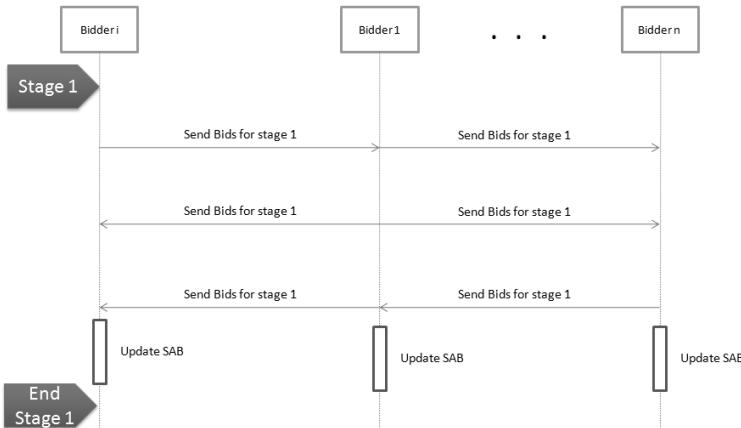


Fig. 2. Messages of the first stage of the PAUSETA protocol

Second and Successive Stages. The second stage and beyond behave in the same way. Figure 3 shows the exchanged messages between the bidders in each stage k of the PAUSETA protocol, for all $k \geq 2$. In these stages, the bidders must make CCBs. The calculation of the CCB involves a combinatorial problem to be solved, which requires a high computation time. Therefore, the bidders use a suboptimal algorithm to calculate it. This algorithm is based on a greedy paradigm. The algorithm builds up a list with all the possible bids. These possible bids are the private ones, subject to the limitation of size given by the current stage k , and the ones stored in the SAB. Then, it sorts this list by adjusting the weight of each possible bid with regard to its assessment and its size of package. Specifically, the weight of each bid is the assessment of the bid divided by the root of the number of resources in the package, since it has been shown experimentally that gives good results [16]. Next, the algorithm selects the bid for the CCB from this sorted list in order of appearance. The bid chosen is valid if:

1. The resources of the bid are required i.e, the algorithm checks that nobody bids for unnecessary resources.
2. The resources have not been allocated yet in other parts of CCB.
3. The bidder has enough resources. A resource can be in several bids due to that the bidders can bid on packages of different size.

In each round within a stage k , $k \geq 2$, each bidder tries to outperform the current best CCB making a new CCB. If it exists, it is broadcasted to the rest of the bidders. If a bidder can not submit a CCB with a higher value than the current winning CCB, it sends a *Failure Bid (FB)*. A FB is a set of bids like a CCB except that it does not participate in the calculation of the winning CCB. The function of FB is to reveal some valuations of bidders, which can be used by other bidders for composing their future CCBs. The FB helps minimize

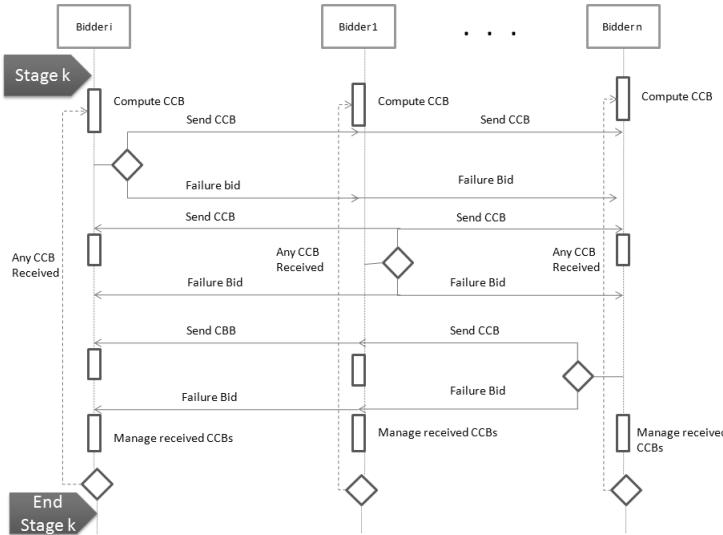


Fig. 3. Messages of the k stage when ($k > 1$)

the threshold problem [17]. This problem occurs when a bidder makes a high bid for a CCB in the first instance, all other bidders must be able to show their assessments to coordinate together and make a higher CCB than the first bidder. After sending the CCB, each bidder computes the highest CCB among all received CCBs storing it as the new current winning CCB. Subsequently, each bidder updates his SAB with the bids of CCB and FB that it has received, and a new round begins. If CCBs have not been received in a round, i.e. only FBs have been received, the current stage ends. The winning CCB of the stage is the winning CCB of the last round. If the stage $k = t$, the auction ends and the winning CCB of the last stage is the winning CCB of the auction.

2.2 Traffic Incidents Ontology Domain

The definition of an ontology facilitates the understanding of the interchanged data shared among different agencies that are involved in the resolution of a traffic incident. This ontology is divided into three subdomains. The environment subdomain, the emergency subdomain and the negotiation subdomain.

The Environment Subdomain. It represents the environment where the traffic incident occurs and where the TMP is implemented. It is a road network in which agencies, resources and the traffic incident are located. This subdomain is shown in Fig. 4 and consists of the following items: 1) road network that represents the physical environment, 2) the location that represents a specific point of the environment and 3) the itinerary that represents a route or path.

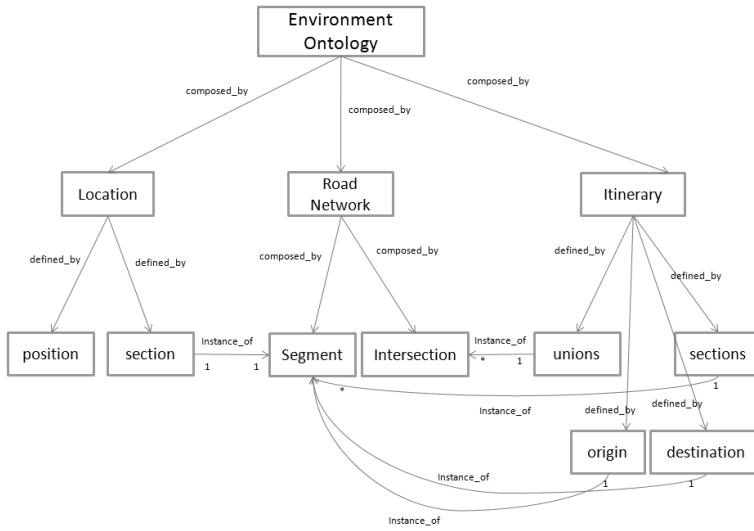


Fig. 4. Environment Ontology

The Road Network. The objects of the road network, Fig. 5, are subdivided into:

1. Segment: Represent a section of a road in a single direction. This section is only accessible from its origin and can only be left by its destination. A segment has the following parameters:
 - (a) The number of lanes that have the road section represented by this segment.
 - (b) The origin is the intersection through which the segment is accessed.
 - (c) The destination is the intersection through which one can exit the segment.
 - (d) The length from the origin to the destination in kilometres.
2. Intersection: it represents the union of several segments. A roundabout, a crossroad, a change of route ... An intersection is defined by the following parameters:
 - (a) The capacity represents the amount of traffic that the intersection can withstand in a given time.
 - (b) The output segments represent the collection of segments that can be accessed from the intersection, i.e. the segments that have as origin this intersection.
 - (c) The input segments represent the collection of segments that give access to a particular intersection, i.e. the segments that have as destination a particular intersection.

The Itinerary. An itinerary is composed by segments united by the corresponding intersections. An itinerary has the following parameters:

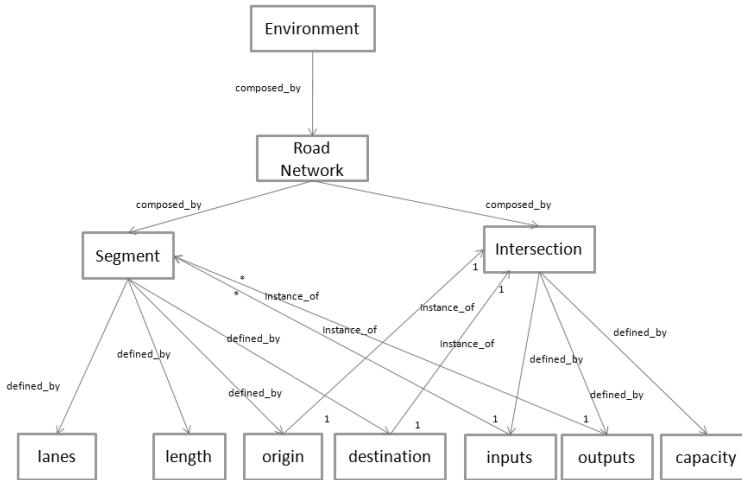


Fig. 5. Environment ontology

1. The sections are all segments that make up the route, road or path.
2. The unions represent the set of intersections that connect the segments which make up the itinerary.
3. The origin is an intersection that represents the point where the itinerary begins.
4. The destination is an intersection that represents the point where the itinerary ends.

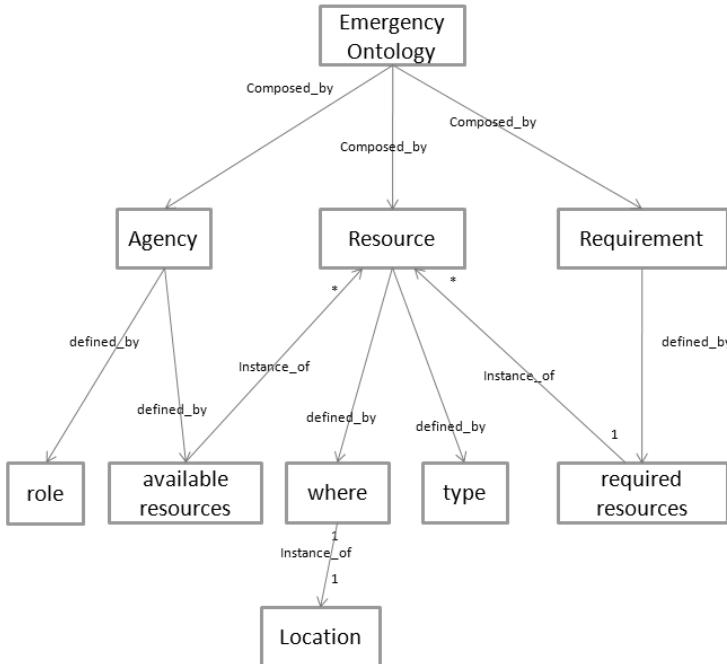
The Location. A location is a specific point in the road network. It has the following parameters:

1. The section is the segment where the location is.
2. The position within the section is where the location is. It is measured in kilometres from the origin of the segment and it can not exceed the length of the segment.

Emergency Ontology Subdomain. It describes the model of emergency. The objects which form part of the treatment of traffic incident are described in this subdomain. See Fig. 6.

The objects which form part of emergency ontology are:

1. Resource: it is an entity that has its own characteristics and it is useful for an emergency. It is composed by the following parameters: 1) what type of resources it is and 2) where it is located. All resources belong to agencies.
2. Agency: it is a private or public company that possesses several resources. It has de following parameters: 1) the resources that the agency posses and 2) the role that it plays. An agency has two roles that are described in Sect. 3.1.

**Fig. 6.** Emergency Ontology

3. Requirement: It is a set of one or more units of one or more type of resources in one or more locations. It represents the resources needed to attend a traffic incident.

Negotiation Ontology Subdomain. It describes the model of negotiation, i.e. the concepts needed to understand and use PAUSETA protocol. The PAUSETA protocol is applied to negotiate what resource has to allocate each agency (Emergency ontology subdomain) and where they should be allocated (Environment Subdomain)

The objects which form part of the negotiation are described in this subdomain. See Fig. 7.

1. Bid: It is an object composed by: 1) the bidder that makes it that is an agency, 2) a package is a set of one or more resources, each of them is different in type, 3) the valuation. Each agency has its own resource valuation function. An agency takes into account the following parameters to assess a resource: the amount of resources available, the route the resource must follow to get to its destination, synergy among resources of the package and the value of the resource itself. The valuation represents the interest that an agency has to assign a package of resources.

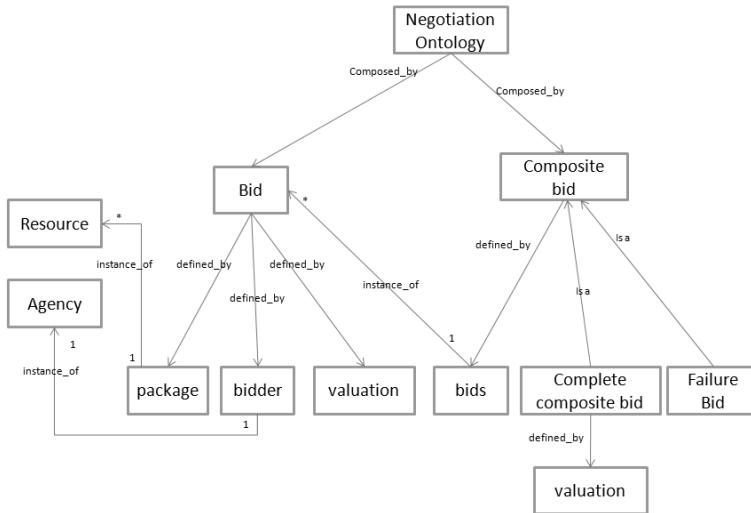


Fig. 7. Negotiation Ontology

2. Composite bid(CB): It is a bid composed by one or more bids. The CB is performed by a bidder. However, the bids that compose it can be performed by several bidders.
3. Complete composite bid(CCB): it is a composite bid that contains bids for all resources defined by the requirements. Moreover, it has a valuation equal to the sum of all valuation of bids that compose it.
4. Failure bid is a CCB that has no valuation. Therefore, it is not involved in the calculation of the winning CCB.

3 Multi-agent Architecture

In this section, the architecture of multi-agent system to attend traffic incidents is described (Fig. 8)

3.1 Negotiating Agents

Each of these agents represents an organism that provides resources. The available information of resources of each agent is stored in its local database, which may have a very different structure from each other. However, the agencies understand a common ontology that allows them to coordinate to perform an activity; in this case, negotiate allocations resources for traffic incidents. Therefore, each of these agents must have its own wrapper for adapting the local data to the ontology described in the previous section. An agent can play two different roles: promoter or cooperator. The cooperator is a negotiating agent who is invited to participate in the negotiation. It has the ability to decide to participate or

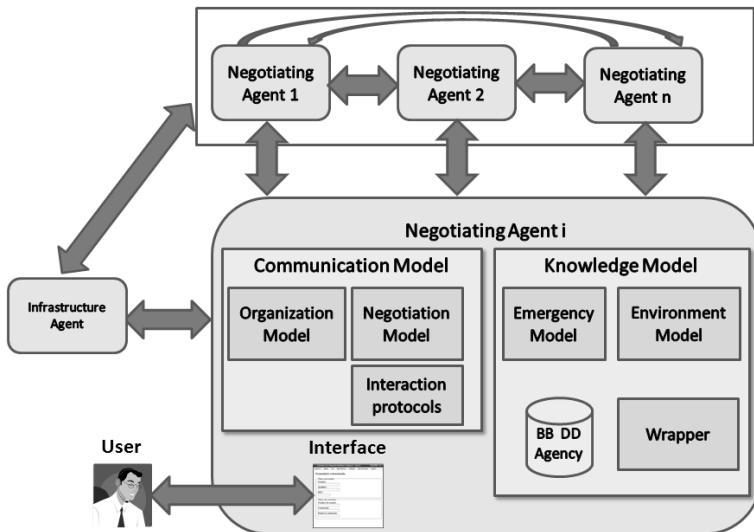


Fig. 8. Architecture of multi-agent system that is responsible for negotiating the resources to attend a traffic incident

not. The promoter is a negotiating agent which receives the information of the required resources to implement a TMP for a traffic incident. Such resources have been provided by the user. This agency is responsible for notifying the co-operator agencies the required resources to address a traffic incident. Once the required resources are reported, the negotiation starts and this agent behaves as a cooperator agent. Both promoter and cooperator implement the PAUSETA protocol as mechanism of negotiation in which each agency is a bidder.

3.2 Infrastructure Agent

The infrastructure provides a yellow pages service by means of which an agent can register, unsubscribe and search for other agents or services in the MAS environment.

3.3 Knowledge Model

This model represents the knowledge that an agent has. It is composed by the following: 1) The emergency model represents the knowledge of the emergency ontology described in Sect. 2.2, 2) the environment model represents the knowledge of the environment ontology described in Sect. 2.2, 3) the Wrapper adapts the local data to the ontologies mentioned and 4) the BBDD is the local data storage.

3.4 Communication Model

This model is composed by organization model, negotiation model and the interaction protocols. The negotiation model represents the negotiation ontology described in Sect. 2.2. The rest of components are described below.

Organization Model. A negotiating agent performs the following procedure to know how many available agents there are in the environment. Whenever an agent is created on the system, it has to be resisted in the infrastructure agent. The negotiating agent that adopts the role of promoter consults the infrastructure agent how many negotiating agents there are available, these adopt the cooperator role. When the promoter receives the response of the infrastructure agent, it contacts with all cooperator agents by sending the requirements. Then, each of these cooperator agents also consult the infrastructure agent to know the rest of the available negotiating agents. Note that the promoter agent does not send the list of available negotiating agents to the rest of them, since it could manipulate the negotiation removing from the list potential competitors. From now on, a peer-to-peer communication in which each agency has a link to all others is created, causing a completely decentralized interaction among the agents.

Interaction Protocols. The interaction protocols define how the agents can communicate with each other. These protocols have been implemented using JADE [18] library. The messages that are exchanged by the agents are divided in two main groups. The messages exchanged in the previous phase of the negotiation and the ones used in the negotiation itself. The negotiation messages have been described in the Sect. 2.1. In the initiation phase of the negotiation, the agent i adopts the role of promoter and it knows the requirements to address the traffic incident. The promoter sends the requirements to the rest of cooperator agents. The cooperator agents can evaluate whether or not to participate in the auction and every one sends its response to all others.

4 Test Case Example

The multi-agent system designed has been implemented to evaluate its behaviour in the final step of a traffic incident resolution. Once the incident has been identified and a TMP has been selected, the final decision on which agency must allocate which resources is decided autonomously and distributed among agencies.

The access to the harbour of Castellón city (Spain) has been modelled with real traffic data. The model includes:

1. The main road network involved. It includes roads CS-22, N-340, CV-183 and CV-1520. There are two neighbour cities in the area, Castellón and Almazora. For the resources located initially in both cities, only distances

to the final allocations have been taken into account (urban streets have not been modelled).

2. A Traffic Management Plan (TMP) for emergencies with 1 scenario, 3 measures and 9 actions.
 3. A set of bidders to model the different agencies involved in incident resolutions:
 - (a) Three Central traffic police stations (with national, regional and local competences).
 - (b) Two crane stations.
 - (c) Two fire stations.
 - (d) Three hospitals.
 - (e) Two civil work stations.
 - (f) A varying number of traffic police units (from whatever central police station) patrolling on the road.

The test consists on the simulation of an incident in the CS-22 motorway. A multiple traffic crash has taken place on the motorway leading to the harbour of Castellón. In the incident, there are two trucks and several cars involved. The loading of the trucks have been spread on the roadway closing the motorway to the harbour of Castellón city. There are 6 injured people, 2 of them are trapped inside the vehicle.

The measures set by the chosen TMP to be deployed for this type of traffic incident are: 1) to evacuate and care the injured people, 2) to clean the incident area and 3) to manage an alternative route to CV-118. Figure 9 presents the modelled area.

The resources to be deployed for dealing with the previous measures are: 4 ambulances; 1 quick response ambulance; 2 urgency physicians; 1 heavy crane; 2 common cranes; 2 fire trucks; 1 maintenance road unit; and 8 traffic police

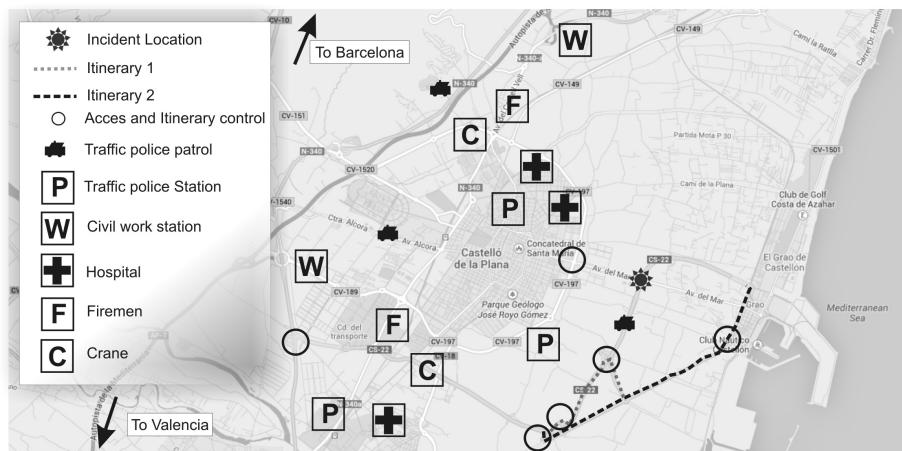


Fig. 9. Real environment modelled: the access to the harbour of Castellón city.

units, 2 to go to the location of the incident and each one of the other 6 going to different places to direct the traffic of the alternative itineraries.

5 Evaluation of the System to Attend Traffic Incidents

A set of tests were performed to evaluate the robustness, scalability and quality of the designed cooperative information system. It is called Multi-agent System that implemented PAUSETA protocol (MSP). All tests were performed in the environment described in the previous section.

5.1 Test 1

In this test, the quality of the results provided by MSP in terms of distance travelled by the resources from where they are currently located to where they should be assigned are evaluated. The greater the distance, the greater the required time. It allows to approximate the response time.

A centralized optimal algorithm in terms of distance has been implemented to carry out this test. This algorithm named Sort Common Path (SCP) minimizes the sum of the distance covered by all resources. The SCP is implemented by a branching and pruning algorithm, where each node of the tree is a resource, and each branch from the root to a leaf is a possible solution. The SCP algorithm assesses each node of each branch. If a node does not drive a better solution than the current best solution, it prunes the branch. When the SCP assesses the left node, it compares if the current solution is better than the current best one. The SCP system provides a lower bound value that a real practical system would never reach because the SCP does not take into account the competences, the private interests of the agencies nor the negotiation time. In this test 100 simulations have been carried out for both systems, MSP and SCP. In each simulation, traffic patrols are randomly placed in the environment.

As shown in Fig. 10, the solution provided by MSP is slightly worse, as expected, but always proportional to the solution provided by the SCP. Each peak and valley are reached simultaneously in both systems. This indicates that, even taking into account the competencies and private interests of agencies, the MSP shows a similar behaviour the SCP.

5.2 Test 2

In this test, the robustness of MSP is evaluated, i.e. the capacity of the system to provide a solution when the agents are not sure that the required resources are available. Environment is configured with different amounts of hospitals keeping the remaining agencies fixed. The number of hospitals in the area starts in 5 and decreases to 1. For each decrease, the average of messages sent by all agencies on all possible scenarios by combining the 5 hospitals is calculated, i.e. if we have the case in which there are 3 hospitals, the average of the messages sent by all agencies is obtained on 10 scenarios, $C_{5,3}$. For each amount of hospitals,

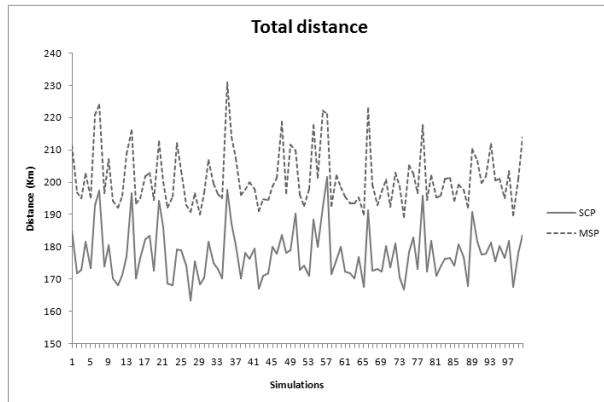


Fig. 10. 100 simulations done for both system: SMC and MSP. Distance measured in km.

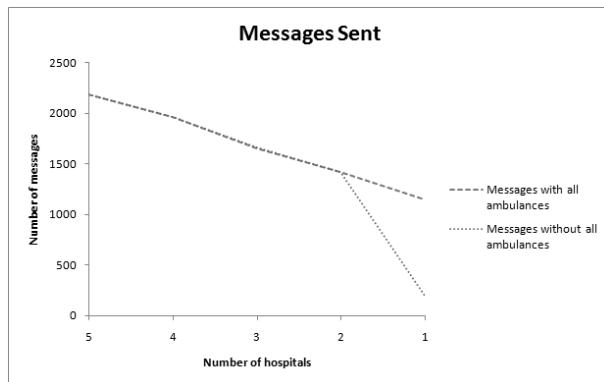


Fig. 11. Messages sent by all the agencies according to the hospitals and their ambulances available in the environment

two tests have been carried out: 1) every hospital has all necessary ambulances to attend the traffic incident by itself, and 2) there are no hospitals that have enough ambulances to attend the traffic incident by itself. The results of this test are shown in Fig. 11.

The number of messages sent between agencies decreases as agencies do so as expected. The MSP provides a solution as long as the agencies have enough resources among all to treat the traffic incident. In addition, although the agencies do not have enough resources to attend the traffic incident by themselves, they are able to reach agreement and provide a solution. In case that there are

insufficient resources to attend the traffic incident, the solution provided in the first stage of the negotiation can be used to know an approximate allocation of resources.

5.3 Test 3

In the last test, the scalability of the MSP is evaluated. For it, a stress test in which all agencies have the required resources to treat the traffic incident by itself has been done. In this test, the amount of agencies in the area is increased. For each increment, 100 simulations have been performed. In each of this simulation, the location of the agencies is randomly made.

Figure 12 shows that the number of messages increases as agencies also do so, i.e. the negotiation intensifies because all agencies want to allocate resources. Although the number of messages increases exponentially, the execution time is linear according to the increase of the agencies as shown in Fig. 13. It is because all agents have to send each message to all others in each round of each stage of the auction. However, the response time is linear because the calculation of the CCBs is performed by a greedy algorithm. This algorithm is linear with respect to the number of bids that manages an agent and in each increment of the agents, a agent only have to take in account bids of the one agent more.

Moreover, the agencies always reach an agreement, in contrast to manual negotiation in which an agreement is not guaranteed.

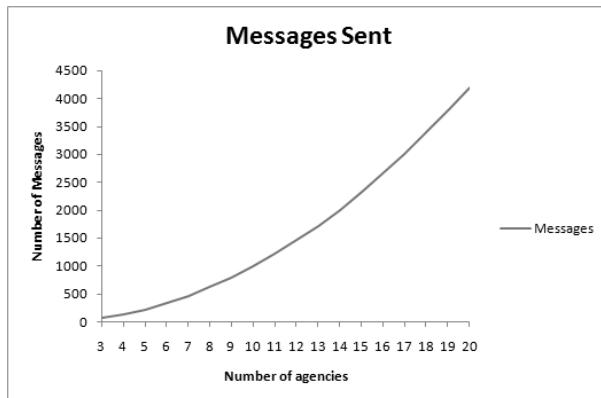


Fig. 12. Average of messages sent by all the agencies according to their increased numbers

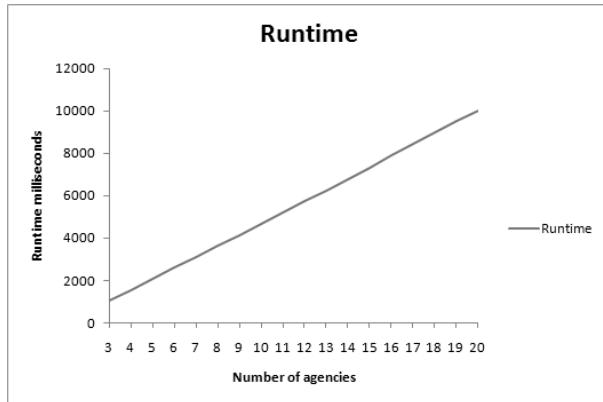


Fig. 13. Runtime of MSP according to the increase of agencies

6 Conclusion

To deal with a traffic incident, several resources must be allocated. These resources are provided by different individual agencies which possess their own interests and different competences. These agencies need to cooperate to decide which one provides each resource. In this paper, a cooperative information system to deal with traffic incidents has been developed. Its aim is to automate the negotiation process. This system has been implemented and evaluated in a real environment. The results show that the implemented system is robust and scalable in different settings of the scenario. Moreover, it is able to give quality solutions taking into account the private interest and the competencies of the agencies. This is due to the negotiation mechanism carried out by the PAUSETA protocol. This protocol allows the agencies to share data that are authorized to be provided. The automation of the process of negation taking into consideration the competences and private interest of the agencies improves the response in respect of manual negotiation. Furthermore, the deployed system always guarantees an agreement among the agencies whenever there are enough resources in the environment to deal with the traffic incidents.

Nowadays, we are working in a dual negotiation protocol to give priority to certain resources in which the response time is more important, for example the ambulances. In addition, we are also working in giving the capacity to the protocol to be able to provide a suitable result in scenarios in which there are not enough resources.

Acknowledgment. This work has been supported by a Universitat Jaume I-Fundació Caixa-Castello research project number P11B2011-46.

References

1. Newgard, C.D., Schmicker, R.H., Hedges, J.R., Trickett, J.P., Davis, D.P., Bulger, E.M., Aufderheide, T.P., Minei, J.P., Hata, J.S., Gubler, K.D., et al.: Emergency medical services intervals and survival in trauma: assessment of the “golden hour” in a north american prospective cohort. *Annals of Emergency Medicine* 55(3), 235–246 (2010)
2. EasyWay Project: DG-TMS-DG07. Traffic Management Plans for Corridors and Networks. Deployment guideline (November 2012)
3. Fernandez, F., Berriochoa, L., Ortega, F.: Methodology for the creation of traffic management plans in case of snow fall. *Dirección General de Tráfico* (2001)
4. Tomás, V.R., García, L.A.: A cooperative multiagent system for traffic management and control. In: V International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2005. ACM (2005)
5. Belda, E., Tomás, V.R.: ITS for emergency resolution. In: Proceedings of 11th World Congress on Intelligent Transport Systems, ITS Japan, Nagoya, Japan, (2004)
6. Cramton, P., Shoham, Y., Steinberg, R.: Combinatorial auctions (2006)
7. Courcoubetis, C., Dramitinos, M.P., Stamoulis, G.D.: An auction mechanism for bandwidth allocation over paths. In: 17th International Teletraffic Congress (ITC), pp. 1–11 (2001)
8. Block, C., Neumann, D., Weinhardt, C.: A market mechanism for energy allocation in micro-chp grids. In: Proceedings of the 41st Annual: Hawaii International Conference on System Sciences, pp. 172–172. IEEE (2008)
9. Rassenti, S.J., Smith, V.L., Bulfin, R.L.: A combinatorial auction mechanism for airport time slot allocation. *The Bell Journal of Economics*, 402–417 (1982)
10. Caplice, C., Sheffi, Y.: Combinatorial auctions for truckload transportation. *Combinatorial Auctions* 21, 539–572 (2006)
11. Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghhraby, W., Griffin, P., Kleywegt, A.: Robot exploration with combinatorial auctions. In: Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 2, pp. 1957–1962. IEEE (2003)
12. De Vries, S., Vohra, R.V.: Combinatorial auctions: A survey. *INFORMS Journal on Computing* 15(3), 284–309 (2003)
13. Farron, M.P., García, L.A., Tomás, V.R., Capdevila, M.: A distributed and automatic negotiation protocol for resource allocation in the management of traffic accidents. In: Proceedings of Systems, Man, and Cybernetics, SMC 2013 (2013)
14. Kelly, F., Steinberg, R.: A combinatorial auction with multiple winners for universal service. *Management Science* 46(4), 586–596 (2000)
15. De Michelis, G., Dubois, E., Jarke, M., Matthes, F., Mylopoulos, J., Papazoglou, M., Pohl, K., Schmidt, J., Woo, C., Yu, E.: Cooperative information systems: a manifesto (1997)
16. Mendoza, B., Vidal, J.M.: On bidding algorithms for a distributed combinatorial auction. *Multiagent and Grid Systems* 7(2), 73–94 (2011)
17. Levin, D.: A simple approach to easing threshold problems in simultaneous auctions with combined value bidding: Contingent bids. *Comments on DA-1075* (2000), <http://wireless.fcc.gov/auctions/31/releases/dlevin.pdf>
18. Bellifemine, F., Poggi, A., Rimassa, G.: Jade—a fipa-compliant agent framework. In: Proceedings of PAAM, London, vol. 99, p. 33 (1999)

Mining Business Process Deviance: A Quest for Accuracy

Hoang Nguyen¹, Marlon Dumas², Marcello La Rosa^{1,3},
Fabrizio Maria Maggi², and Suriadi Suriadi¹

¹ Queensland University of Technology, Australia

{huanghuy.nguyen@student.,m.larosa,s.suriadi}@qut.edu.au

² University of Tartu, Estonia

{marlon.dumas,f.m.maggi}@ut.ee

³ NICTA Queensland Lab, Australia

Abstract. This paper evaluates the suitability of sequence classification techniques for analyzing deviant business process executions based on event logs. Deviant process executions are those that deviate in a negative or positive way with respect to normative or desirable outcomes, such as executions that undershoot or exceed performance targets. We evaluate a range of features and classification methods based on their ability to accurately discriminate between normal and deviant executions. We also analyze the ability of the discovered rules to explain potential causes of observed deviances. The evaluation shows that feature types extracted using pattern mining techniques only slightly outperform those based on individual activity frequency. It also suggests that more complex feature types ought to be explored to achieve higher levels of accuracy.

1 Introduction

Process mining is a family of techniques to extract knowledge of business processes from event logs [13]. It encompasses, among others, techniques for automated discovery of process models from logs, techniques for checking conformance between a given process model and an event log, as well as techniques for analyzing and predicting performance of business processes based on event logs.

This paper deals with *business process deviance mining*, a family of techniques aimed at analyzing event logs in order to explain the reasons why a business process deviates from its normal or expected execution. Such deviations may be of a negative or of a positive nature. Positive deviance corresponds to executions that lead to high process performance, such as achieving positive outcomes with low execution times, low resource usage or low costs. Negative deviance refers to the executions of the process with low process performance or with negative outcomes (e.g. customer complaints) or compliance violations.

The input of business process deviance mining is a set of labelled traces. Each trace represents one execution (case) of the process under analysis. Each trace is a sequence of events, wherein each event records the execution of an activity. The label associated with a trace indicates whether it is normal or deviant. Given this

input, the problem of deviance mining is to compute a function (called a *classifier*) that takes as input a trace and outputs its class (normal or deviant). Such function must produce accurate labels, i.e. it should guess the correct class of a trace both for traces in the input (training) set but also for other unseen traces. In addition, as the purpose of deviance mining is explanatory, the function must be captured in terms of patterns or rules interpretable by an analyst.

A family of techniques applicable to deviance mining is *sequence classification* [14], where the goal is to build classifiers that discriminate between two or more classes of sequences. One key step in sequence classification is to extract features from sequences that can be given as input to standard classification techniques such as decision trees. Such features are typically extracted using sequence mining techniques. Various such techniques have been independently tested in the context of deviance mining as discussed later. However, no comparative study has been conducted to assess their relative merits in this setting.

This paper presents a comparative evaluation of sequence mining techniques for business process deviance mining. The paper compares two families of techniques: frequent pattern mining and discriminative mining. Techniques are benchmarked using a battery of event logs covering situations where deviance is frequent (balanced datasets) and others where deviance is rare (unbalanced).

The paper is structured as follows. Section 2 discusses existing deviance mining methods. Section 3 outlines the methods for feature extraction and classification evaluated in this study. Next, Section 4 presents the comparative evaluation. Section 5 summarizes the contribution and discusses directions for future work.

2 Related Work

Business process deviance mining has been the subject of many case studies where a variety of techniques have been applied. For example, in [11] we report on a case study in an insurance company aimed at explaining why some simple claims took too long to be resolved. We applied a technique called *delta-analysis*, which consists in using automated process discovery to extract two process models: one for “normal” cases and one for “deviant” cases, and manually comparing these two models. Delta analysis has also been applied to explain deviance in healthcare processes [8]. In this paper, we do not consider delta analysis because of its manual nature. Instead we focus on automated techniques.

In [10], the authors analyzed a log of a software defect handling process to discriminate between defect reports leading to correct resolution (normal) vs. those leading to complaints (deviant). They applied a *discriminative pattern mining* algorithm to identify patterns of the form “activity *B* occurs *N* times after activity *A* has occurred *M* times”, that are frequent in deviant cases but not in normal cases or vice-versa. A decision tree is built based on these features.

In [2], the authors sought to discriminate between traces leading to malfunctioning (versus normal functioning) of X-ray machines. Unlike [10] where discriminative sequence mining was employed, [2] employs *frequent pattern mining* to extract so-called “tandem repeats”, “maximal repeats” and “alphabet repeats” [3]. A tandem repeat is a sequence of events that is repeated; a maximal

repeat in a log is a sequence of events that is repeated and not included in a longer repeated sequence; a repeat alphabet is any non-empty intersection of the set of events contained in different tandem/maximal repeats. In control-flow terms, tandem repeats correspond to loops, maximal repeats to subprocesses, and alphabet repeats to parallelism. In [2], tandem repeats, maximal repeats and alphabet repeats are extracted for all traces combined (normal and deviant). The patterns with highest support are then used to build a decision tree.

Similarly, in [4] the authors applied frequent pattern mining to discriminate between cases with positive clinical outcomes (vs. negative ones) in a process for congestive heart failure treatment. The authors extracted frequent patterns of the form “ B occurs after A ” from positive cases and from negative cases separately. The extracted patterns were used together with manual delta-analysis to extract pathways characteristic of either positive or negative cases.

[12] presents a case study where analysts in a company sought to identify causes for non-compliant cases in a procurement process. The authors applied association rule mining to extract frequent patterns for normal and for deviant cases separately. The patterns were used to derive rules characterizing deviance.

In the above studies, the input are sequences of activity occurrences without payload. Sometimes, events in the log carry a payload such as attributes representing resources, or attributes provided as input to the process (e.g. customer type, age) or produced in the process. Preliminary studies have attempted to use such data for deviance mining [6,9]. In this paper, we focus on logs consisting of sequences of events without payload. This means we make minimal assumptions on the log and study how much accuracy can be achieved in this setting.

3 Model Construction

All automated deviance mining techniques reviewed above involve two phases: (i) a pattern extraction phase where patterns are extracted from traces seen as sequences of symbols representing activity occurrences; and (ii) a classification phase, where each trace is abstracted as a vector of features, and these vectors are used to produce a classifier. The reviewed techniques differ on the employed pattern extraction technique and/or the employed classification method. From the feature extraction perspective, we evaluate the following methods:

1. Occurrence count of *individual activities* in a trace [11]. In this feature extraction method, each activity type (e.g. “Issue Invoice”, “Invoice Paid”) becomes a numerical feature. For a given trace t , the value of an activity’s feature for t is the number of times the activity occurs in t .
2. *Tandem repeats (TR)*, *alphabet tandem repeats (ATR)*, *maximal repeats (MR)* and *alphabet maximal repeats (AMR)* [2]. Here, each pattern (tandem repeat, maximal repeat, etc.) is a boolean feature. For a given trace, the feature corresponding to a repeat pattern is true iff the pattern occurs in the trace. We select for evaluation the repeat patterns defined in [3] as they

capture common control-flow relations (loops, parallelism, subprocesses) and have been shown to be suitable for deviance mining [2]. Given the potentially large number of TR/MR patterns that can be extracted from a log, we select N with highest support where N is a parameter of the method.

3. *Discriminative (iterative) patterns (DP)* [5]. Here, the features are iterative patterns (sequences of consecutive events) that appear many times within a trace but also across traces. The N iterative patterns with the highest discriminative power (measured via the *Fischer score*) become boolean features. Among existing discriminative sequence mining techniques, we select the one in [5] because it falls in the same category as the one in [10] and it is comparable to the technique of [2], as tandem repeats are iterative patterns that occur frequently within and across traces.

The first method is a baseline. Our aim is to evaluate the added-value of other feature extraction methods (frequent and discriminative patterns) over *individual activities*. Thus, we study 6 feature sets: (i) individual activities (IA); (ii) IA+TR; (iii) IA + ATR; (iv) IA + MR; (v) IA + AMR; and (vi) IA + DP.

Given a set of features and a labelled log (i.e. a set of traces labelled as normal/deviant), the reviewed techniques extract a labeled sample ($\langle f_1, \dots, f_n \rangle, l$) for each trace t , where f_i is the value of the i^{th} feature for t and l is the label (normal/deviant). A range of methods can be used to construct classifiers from labelled samples. Since we seek interpretable classifiers, a natural choice are decision trees as in [2, 10, 11]. We specifically use the C4.5 method in RapidMiner. We also include k-NN (k-Nearest Neighbors) in the evaluation. In this method, a sample is assigned the most common label among the sample's k nearest neighbors in the training set. We set $k = 8$ after trial-and-error to find a value yielding highest accuracy. Although no rules can be extracted from a k-NN classifier, its output is explainable since given a trace t , one can show which similar traces have been used to classify t . Finally, we included neural networks in the evaluation as representative of a method that can adjust itself to the data and handle large feature sets [15] although it does not produce interpretable rules.

Table 1. Descriptive statistics for the six datasets, including whether the distribution of normal (norm.) and deviant (dev.) cases is balanced, and whether the deviance criterion used is temporal (temp.) or non-temporal (non-temp.).

Dataset	Normal cases	Deviant cases	Total cases	Avg. length (norm.)	Avg. length (dev.)	Event classes (norm.)	Event classes (dev.)	Is balanced?	De-viance criterion
Hospital	448	363	811	16	20	25	23	Yes	temp.
Insurance	1,921	3,195	5,116	13	24	13	12	No	temp.
BPI _{dCC}	917	225	1,142	109	85	100	100	No	non-temp.
BPI _{dM13}	832	310	1,142	144	74	99	99	No	non-temp.
BPI _{dM16}	926	216	1,142	127	113	99	94	No	non-temp.
BPI _{t101}	683	459	1,142	195	25	107	103	Yes	non-temp.

4 Evaluation

4.1 Datasets

We selected six real-life datasets in order to cover a range of process deviance types. Specifically, these datasets contain cases labelled as “normal” and “deviant” based either on *temporal* or *non-temporal* criteria. The former differentiates between normal and deviant cases based on process duration w.r.t. a threshold (e.g. slow cases if above 180 min); the latter criterion differentiates cases based on data attributes (e.g. patient suffered from a given cancer or not). A further dimension for the selection of the datasets was the distribution of deviant cases vs. normal cases (balanced or unbalanced).

The first dataset (called Hospital in Table 1) records the flow of chest pain patients in an emergency department of an Australian hospital. Each case is labelled as “quick” if completed within 180 min and “slow” if completed over 180 min. Thus, we used a temporal criterion to classify this log. The second log (called Insurance) comes from a large Australian insurance company and records an extract of the instances of a commercial insurance claims handling process executed in 2012 [11]. The temporal criterion is 30 days. The remaining four datasets were extracted from the BPI challenge 2011 log [1]. This log records the executions of a process related to the treatment of patients diagnosed with cancer in a Dutch hospital. The log contains domain specific case attributes, e.g. *Age*, *Diagnosis*, *Diagnosis code*, and *Treatment code*. We extracted four logs according to four deviance criteria: (i) Deviant cases if Diagnosis is “cervix cancer” (BPI_{dCC} dataset in Table 1); (ii) Deviant cases if Diagnosis code is “M13” (BPI_{dM13}); (iii) Deviant cases if Diagnosis code is “M16” (BPI_{dM16}); (iv) Deviant cases if Treatment code is “101” (BPI_{t101}).

4.2 Classification Accuracy

We measured classification accuracy using the standard notion of *accuracy* defined as $\frac{tp+tn}{tp+tn+fp+fn}$ where tp and tn are the # traces correctly classified as deviant and normal (true positives and true negatives), fp is the # false positives and fn is the # false negatives. We also report on the *Area Under the ROC Curve (AUC)* of each classifier. This corresponds to the probability that a random negative sample is ranked higher than a random positive sample in the list of samples ranked from most to least likely to belong to the deviant class.

In order to test all combinations of feature types and classification methods for all six datasets under exam, we created a scientific workflow in RapidMiner v6.0. The vector spaces containing the extracted features were stored in a MS Access database. Tandem/maximal repeats and their alphabet variants were extracted using the ProM plugin described in [2], while discriminative patterns were extracted using the tool implementation in [5].¹

¹ The BPI log extracts, the scientific workflow and the results of the tests can be downloaded from <http://tinyurl.com/kvqtep>

Tables 2–7 show the results of the measurements of the classification accuracy for the six datasets using the feature types and classification methods discussed in Section 3. The tables report mean accuracy and AUC obtained for each classification method based on five-fold cross-validation, meaning that each dataset is split five times into 80% of the dataset for training and 20% for testing and accuracy/AUC is calculated for each such “fold” and aggregated across all five folds. Next to accuracy we also show the interval of accuracy/AUC values across all folds in the form of a $+/-\delta$ bracket from the mean (standard deviation).

Table 2. Classification Results for Hospital dataset (temporal – balanced)

Feature type	Decision Tree		k-NN		Neural Net	
	AC(%)	AUC	AC(%)	AUC	AC(%)	AUC
IA	66.09 \pm 1.53	0.683 \pm 0.008	70.66 \pm 3.25	0.761 \pm 0.038	69.55 \pm 2.78	0.751 \pm 0.039
IA+TR	65.35 \pm 2.45	0.639 \pm 0.034	69.67 \pm 3.16	0.769 \pm 0.031	66.22 \pm 3.74	0.714 \pm 0.049
IA+ATR	68.19 \pm 3.69	0.689 \pm 0.043	70.04 \pm 2.99	0.766 \pm 0.029	67.69 \pm 1.20	0.741 \pm 0.020
IA+MR	64.37 \pm 3.38	0.627 \pm 0.033	70.16 \pm 1.74	0.764 \pm 0.019	66.59 \pm 3.65	0.735 \pm 0.036
IA+AMR	65.72 \pm 1.66	0.645 \pm 0.027	69.54 \pm 3.23	0.766 \pm 0.036	66.95 \pm 2.93	0.736 \pm 0.036
IA+DP	66.70 \pm 2.73	0.645 \pm 0.038	70.53 \pm 2.66	0.762 \pm 0.026	65.60 \pm 2.65	0.712 \pm 0.045

Table 3. Classification Results for Insurance dataset (temporal – unbalanced)

Feature type	Decision Tree		k-NN		Neural Net	
	AC(%)	AUC	AC(%)	AUC	AC(%)	AUC
IA	83.17 \pm 1.77	0.857 \pm 0.017	83.87 \pm 1.00	0.908 \pm 0.011	86.49 \pm 1.18	0.937 \pm 0.006
IA+TR	83.33 \pm 0.71	0.838 \pm 0.007	83.64 \pm 0.82	0.912 \pm 0.009	83.13 \pm 1.84	0.894 \pm 0.016
IA+ATR	82.60 \pm 1.49	0.832 \pm 0.027	83.80 \pm 1.01	0.912 \pm 0.010	83.97 \pm 2.62	0.895 \pm 0.028
IA+MR	82.62 \pm 0.84	0.825 \pm 0.021	83.91 \pm 1.30	0.912 \pm 0.010	84.46 \pm 0.98	0.903 \pm 0.011
IA+AMR	82.66 \pm 0.92	0.813 \pm 0.023	84.28 \pm 0.92	0.912 \pm 0.012	83.11 \pm 0.89	0.900 \pm 0.010
IA+DP	83.50 \pm 1.16	0.840 \pm 0.017	85.13 \pm 0.72	0.918 \pm 0.010	83.48 \pm 1.21	0.916 \pm 0.003

Table 4. Classification Results for BPI_{dCC} dataset (non-temporal – unbalanced)

Feature type	Decision Tree		k-NN		Neural Net	
	AC(%)	AUC	AC(%)	AUC	AC(%)	AUC
IA	78.81 \pm 2.21	0.752 \pm 0.026	79.95 \pm 1.15	0.751 \pm 0.040	78.37 \pm 3.22	0.771 \pm 0.057
IA+TR	76.97 \pm 4.14	0.736 \pm 0.061	81.17 \pm 2.71	0.761 \pm 0.039	72.78 \pm 12.82	0.724 \pm 0.030
IA+ATR	78.19 \pm 2.54	0.738 \pm 0.061	80.56 \pm 1.90	0.760 \pm 0.025	79.16 \pm 1.22	0.684 \pm 0.035
IA+MR	76.35 \pm 1.93	0.682 \pm 0.050	80.38 \pm 2.10	0.773 \pm 0.020	75.32 \pm 6.93	0.659 \pm 0.112
IA+AMR	75.66 \pm 1.78	0.687 \pm 0.038	80.21 \pm 2.00	0.771 \pm 0.022	79.25 \pm 1.34	0.675 \pm 0.076
IA+DP	78.98 \pm 2.15	0.744 \pm 0.033	80.65 \pm 1.12	0.771 \pm 0.019	78.98 \pm 1.20	0.681 \pm 0.061

From the results, we can draw the following observations. First, when the deviance criterion is temporal, i.e. based on the duration of the process (Hospital and Insurance datasets), IA alone tends to achieve the highest accuracy levels, though the difference w.r.t. other feature types is minimal. For example, in the Hospital dataset, accuracy is $\sim 70\%$ with k-NN across all feature types; in the Insurance dataset, Neural Networks achieve the highest value (86.5%) via IA, with the other feature types/classifiers ranging from 82% to 85%. In the Insurance dataset, we also get the highest AUC with Neural Networks on top of IA,

Table 5. Classification Results for BPI_{dM13} dataset (non-temporal – unbalanced)

Feature type	Decision Tree		k-NN		Neural Net	
	AC(%)	AUC	AC(%)	AUC	AC(%)	AUC
IA	71.63±2.01	0.721±0.033	72.59±0.92	0.700±0.011	71.98±2.49	0.751±0.026
IA+TR	71.19±2.00	0.710±0.042	72.07±1.36	0.705±0.016	72.42±1.15	0.671±0.033
IA+ATR	72.33±1.60	0.691±0.012	71.72±0.78	0.698±0.007	69.08±9.15	0.641±0.101
IA+MR	71.98±2.59	0.722±0.026	71.28±0.91	0.692±0.013	72.24±2.21	0.693±0.050
IA+AMR	72.33±2.97	0.728±0.029	71.19±1.09	0.692±0.013	71.29±5.79	0.662±0.099
IA+DP	73.99±3.33	0.727±0.064	72.85±2.27	0.728±0.045	71.36±2.24	0.694±0.055

Table 6. Classification Results for BPI_{dM16} dataset (non-temporal – unbalanced)

Feature type	Decision Tree		k-NN		Neural Net	
	AC(%)	AUC	AC(%)	AUC	AC(%)	AUC
IA	82.49±1.06	0.759±0.058	83.27±1.81	0.774±0.031	83.45±2.30	0.832±0.058
IA+TR	83.19±0.91	0.763±0.028	83.19±1.17	0.771±0.022	82.75±0.74	0.799±0.030
IA+ATR	83.10±1.35	0.749±0.069	83.10±1.22	0.767±0.025	72.87±14.48	0.759±0.049
IA+MR	82.57±0.94	0.766±0.049	82.84±1.44	0.776±0.026	81.53±1.85	0.773±0.047
IA+AMR	82.57±0.94	0.766±0.051	82.84±1.44	0.776±0.027	79.78±4.27	0.799±0.054
IA+DP	84.06±0.79	0.736±0.076	84.68±0.61	0.803±0.008	82.84±2.24	0.834±0.025

Table 7. Classification Results for BPI_{t101} dataset (non-temporal – balanced)

Feature type	Decision Tree		k-NN		Neural Net	
	AC(%)	AUC	AC(%)	AUC	AC(%)	AUC
IA	84.94±1.76	0.860±0.017	85.99±2.09	0.910±0.026	83.10±4.25	0.893±0.030
IA+TR	84.94±1.66	0.850±0.009	87.57±2.37	0.913±0.026	75.57±8.64	0.728±0.262
IA+ATR	85.38±1.47	0.861±0.013	86.69±2.73	0.911±0.027	69.61±11.63	0.766±0.233
IA+MR	84.76±1.60	0.859±0.017	86.43±3.28	0.912±0.026	67.34±8.97	0.564±0.273
IA+AMR	84.77±1.61	0.855±0.011	86.17±3.39	0.913±0.028	71.19±9.60	0.625±0.297
IA+DP	84.77±2.75	0.864±0.037	87.48±2.03	0.920±0.006	62.08±10.56	0.510±0.255

while in the Hospital dataset, the highest AUC is obtained by k-NN, which is essentially the same across all feature types ($\sim 0.76\%$). These results suggest that IA already carries most of the signal when the labeling of deviance/normal cases is based on a temporal criterion. This is attributable to the fact that process duration is directly correlated with the number of activities being performed, so the more activities are repeated, the longer a process case will take. More precisely, a repeated activity indicates a loop in the process, which is typically symptomatic of process delays. For example, in the case of the Insurance dataset this relates to the repetition, among others, of activity “Request additional information”, indicating that there is no sufficient information to progress the handling of the claim (e.g. further evidence of an accident is needed).

Second, when the deviance criterion is not temporal but based on a data attribute (this is the case in all BPI datasets), we observe a marginal increase of accuracy with sequence mining techniques. In particular, we obtain the highest accuracy with tandem repeats in the BPI_{dCC} (81.2%) and BPI_{t101} (87.6%), and with discriminative patterns in BPI_{dM13} (74%) and BPI_{dM16} (84.7%). These results tend to be confirmed by the AUC, whose highest values are obtained

by a sequence mining technique, though not necessarily the same, e.g. in the BPI_{dCC} dataset the highest AUC (0.77) is achieved by IA + MR whereas the highest accuracy is achieved by IA + TR. An exception is made by the BPI_{dM13} dataset, where the highest AUC is achieved by IA alone (0.75). These results suggest that in the case of a labeling not dependent on process duration, the total number of activities alone is not enough to explain why certain deviances occur. That said, once again, the increase of accuracy achieved by the sequence mining techniques is marginal compared to IA alone, indicating that probably other feature types such as inter-arrival rate, resources and input/output data, have to be extracted to be able to better discriminate between normal and deviant cases. We also remark that none of the techniques indeed achieves an accuracy of 95% or above, with values ranging from 64.4 to 87.6%.

Third, the three classification methods under analysis all produce stable results in terms of standard deviation (relatively low). An exception is made by Neural Networks, which have high standard deviation in all BPI datasets. This is probably due to the type of classification based on a specific data attribute rather than on a temporal criterion. Further, out of all feature types, discriminative patterns tend to have the most stable results (lowest standard deviation) across all datasets and feature types, with an exception being the BPI_{t101} dataset.

4.3 Rules Interestingness

We analyzed the capability of the rules extracted from decision trees to explain as much as possible deviant process executions using the least amount of rules. For each rule, we calculated its **coverage** as the ratio between the number of deviant traces that satisfy a rule and the total number of deviant traces (i.e. the recall), and its **precision** as the ratio between the number of traces that satisfy a rule and the number of traces that are classified correctly. For example, using IA+DP on the insurance log, we obtain the following rule: if the patterns (`New Claim`, `Authorise Payment`, `Close File`) and (`New Claim`, `Close File`) do not occur, the trace is classified as “slow”. The coverage of this rule is 45.23% while its precision is 98.84%.

To compare the overall interestingness of rules through the use of different feature types, we use the ratio between the average coverage of all rules derived from the use of a particular feature type, and the total number of rules. The `avg coverage/#rules` ratio reflects the strength and simplicity of a ruleset (the higher the more powerful, as more deviant cases can be explained with less rules). From the results we observe that the rulesets mined from IA+AMR and IA+DP have the highest `coverage/#rules` ratio. However, neither the coverage nor the `#rules` depend on the chosen feature type; rather, they depend on the dataset characteristics. For detailed results we refer to the technical report [7].

5 Conclusion

Existing methods for business process deviance mining extract patterns from event logs based on frequent or discriminative pattern mining. We have

empirically observed that in processes with high variability, (discriminative) pattern mining approaches may slightly outperform those based purely on activity occurrence counts. However, in all cases, the accuracy obtained with such approaches is limited (rarely above 80%). Underlying this limitation is the fact that the reviewed methods treat the input as sequences of simple symbols representing activity occurrences. Oftentimes, including all six datasets in this study, business process logs consist instead of *temporal complex* symbolic sequences, i.e. sequences of timestamped events with payloads consisting of attribute-value pairs. A direction for future work is to develop and apply techniques for extracting (discriminative) patterns from complex symbolic sequences – a non-trivial and open problem [14]. While tackling the problem of complex symbolic sequence mining in the general case is very challenging, it may be possible to reduce the problem of business process deviance mining to well-scoped subsets of this problem, for example by taking advantage of information contained in available process models in order to prune the pattern search space.

Acknowledgments. Work funded by the Estonian Research Council, ERDF via the Estonian Centre of Excellence Programme and by National ICT Australia.

References

1. 3TU Data Center. BPI Challenge, Event Log (2011), doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
2. Bose, R.P.J.C., van der Aalst, W.M.P.: Discovering signature patterns from event logs. In: Proceedings of CIDM, pp. 111–118. IEEE (2013)
3. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace clustering based on conserved patterns: Towards achieving better process models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBP, vol. 43, pp. 170–181. Springer, Heidelberg (2010)
4. Lakshmanan, G.T., Rozsnyai, S., Wang, F.: Investigating clinical care pathways correlated with outcomes. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 323–338. Springer, Heidelberg (2013)
5. Lo, D., Cheng, H., Han, J., Khoo, S.-C., Sun, C.: Classification of software behaviours for failure detection: A discriminative pattern mining approach. In: Proc. of KDD, pp. 557–566. ACM (2009)
6. Nakatumba, J., van der Aalst, W.M.P.: Analyzing resource behavior using process mining. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBP, vol. 43, pp. 69–80. Springer, Heidelberg (2010)
7. Nguyen, H., Dumas, M., La Rosa, M., Maggi, F.M., Suriadi, S.: Mining business process deviance: A quest for accuracy. ePrint 75279, QUT (2014), <http://eprints.qut.edu.au/75279/>
8. Partington, A., Wynn, M.T., Suriadi, S., Ouyang, C., Karnon, J.: Process mining of clinical processes: Comparative analysis of four australian hospitals. ACM Trans. in Management Information System (in press, 2014)
9. Poelmans, J., Dedene, G., Verheyden, G., Van der Mussele, H., Viaene, S., Peters, E.: Combining business process and data discovery techniques for analyzing and improving integrated care pathways. In: Perner, P. (ed.) ICDM 2010. LNCS, vol. 6171, pp. 505–517. Springer, Heidelberg (2010)

10. Sun, C., Du, J., Chen, N., Khoo, S.-C., Yang, Y.: Mining explicit rules for software process evaluation. In: Proc. of ICSSP, pp. 118–125. ACM (2013)
11. Suriadi, S., Wynn, M.T., Ouyang, C., ter Hofstede, A.H.M., van Dijk, N.J.: Understanding process behaviours in a large insurance company in australia: A case study. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 449–464. Springer, Heidelberg (2013)
12. Swinnen, J., Depaire, B., Jans, M.J., Vanhoof, K.: A process deviation analysis – A case study. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 87–98. Springer, Heidelberg (2012)
13. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
14. Xing, Z., Pei, J., Keogh, E.J.: A brief survey on sequence classification. SIGKDD Explorations 12(1), 40–48 (2010)
15. Zhang, G.P.: Neural networks for classification: A survey. IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews 30(4), 451–462 (2000)

Multi-paradigm Process Mining: Retrieving Better Models by Combining Rules and Sequences

(Short Paper)

Johannes De Smedt*, Jochen De Weerdt, and Jan Vanthienen

KU Leuven Faculty of Economics and Business
Department of Decision Sciences and Information Management
Naamsestraat 69 B-3000 Leuven, Belgium
`firstname.lastname@kuleuven.be`

Abstract. Business process mining is a well-established field of research which focuses on the automatic retrieval and analysis of process flows. The discovery and representation of these models is based on techniques that come in all shapes and forms. Most notably, procedurally-based algorithms such as Heuristics Miner have been used successfully for this purpose. Also, declarative process model miners have been proposed, which give other insights into the model by generating rules that apply on the activities. This paper proposes an integrated approach to combining these paradigms to discover process models that contain best of both worlds to enrich insights into the event logs under scrutiny.

Keywords: Business Process mining, Mixed-Paradigm mining, Causal Nets, Declare.

1 Introduction

Over the last decade, the field of process mining has gained a lot of traction. Its main focus lies on the automatic retrieval and subsequent analysis of business process models and insights from data logs containing events [1]. The three pillars of process mining focus on process discovery, enhancement and conformance checking. The former can be considered the primordial task in a process mining exercise, with the two latter pillars typically building on it. The goal of process discovery is to learn a process model from data in the form of an event log in the most comprehensive, comprehensible and correct way. In order to do so, many mining techniques have been proposed, including, amongst others, Alpha Miner and Heuristics Miner [1]. The representation form used by these models are procedural models.

More recently, declarative process modeling and mining has gained popularity and several discovery techniques such as, e.g., Declare Miner [2] have been

* Corresponding author.

implemented for discovery purposes. These miners retrieve rules from process logs to create models with a more flexible view on the information contained in the log, as any behavior that is not strictly forbidden is allowed.

This paper presents an algorithm for combining procedural and declarative constraints in one map and shows results that indeed provide a new way of looking at mined processes. Both process mining approaches offer different characteristics that each enlightens certain aspects of an event log, but in the literature, the combination of both paradigms into one discovery algorithm has received very little attention so far. Nonetheless, the prospect of learning richer, multi-paradigm process models seems promising, especially in the context of semi- or unstructured processes. Our proposed approach has been implemented in ProM¹ as a plug-in which builds on the two most frequently used process mining techniques for each paradigm, (Flexible) Heuristics Miner and Declare Miner.

The remainder of this paper is structured as follows. The second section covers the related work, followed by a running example which uncovers some issues found for current approaches in section three. Section 4 contains a comparison of and opportunities for combined mining approaches. Next, section five provides an overview of the implementation, followed by an evaluation section with results for the running example. The last section concludes the paper with a discussion and future work.

2 Related Work

Within the field of process mining, a strong emphasis is put on the automatic retrieval of business process models from event logs. Numerous techniques have been proposed, such as Alpha Miner, Heuristics Miner, Fuzzy Miner and a Genetic Miner [1]. These algorithms offer mining solutions that deal with aspects such as the trade-off between recall, precision, generalization, and the presence of noise in the event log. Initially, procedural process mining approaches were put forward. They capture sequence constraints and parallelism by incorporating information supporting adjacency and (direct) succession in a process log, extended with (X)OR- and AND-split and -join information. The techniques represent their outcomes often in process models such as Petri nets [3].

More recently, the modeling and mining of flexible process models has gained popularity among researchers. The most prominent declarative control flow modeling frameworks are DecSerFlow [4] and its successor Declare [5], which offer a set of Linear Temporal Logic (LTL)-based constraint templates for modeling and rule verification purposes, bundled in the ConDec language. Many declarative process discovery algorithms have been developed, such as [2,6,7,8].

A real mixed-paradigm outcome, however, has not been pursued yet as such, with the very recent exception of [9] in which the authors break down the event log into a hierarchy and mine the different subprocesses according to the appropriate paradigm. While the authors propose an approach based on counting

¹ <http://www.processmining.org/>

predecessors and successors which is somewhat comparable to direct succession in Section 4, applying a threshold on the exact number of predecessors and successors seems rather coarse for deciding on the structured versus unstructured nature of activities, especially for smaller logs. Therefore, our approach includes a configurable threshold based on the concept of entropy, which allows for making a more versatile trade-off between structured and unstructured behavior. In addition, the mandatory hierarchical structure of the mined hybrid model puts a limitation on its application to event logs where structured and unstructured behavior are much more intertwined. The approach presented in this paper does not presume such a hierarchical structure. Finally, the visualization in [9] seems strongly disjoint while our technique is capable of representing any mixture of procedural and declarative constraints in a single mined process model.

3 Running Example

As a running example, we provide the simple multi-paradigm process model in Figure 1. Both declarative ConDec constraints and more sequentially-based Petri nets are combined to resemble the progress of a PhD student throughout his career, which contains the strict order of a first and second seminar followed by the defence. Meanwhile, he/she creates content which is subsequently published in journals or presented at a conference, resembled by the *Alternate Precedence* constraints. This constraint expresses that both *Journal Paper* and *Conference* can happen after *Content Creation*, and again only after the next occurrence of the *Content Creation* activity. The first seminar cannot happen before a first contribution to a conference and the second seminar has to be preceded by a journal publication. Note that, while the multi-paradigm model is fairly simple and understandable, the simulated event log presents characteristics that are typically found in real-life logs originating from complex processes.

In order to assess the ability of procedural miners to retrieve the complex, multi-paradigm, and flexible relations between these activities, we enacted the model and mined the simulation log with (Flexible) Heuristic Miner. The algorithm is unable to retrieve the exact position and relation of the three activities *Content Creation*, *Conference* and *Journal Paper* as shown in Figure 2. While Heuristics Miner captures loops and invisible events to support the quite random appearance of *Content Creation*, it fails to capture the relation of *Journal Paper* and *Second Seminar*. The model does not support the *Precedence* constraint and leaves no room to repeat the process after firing *Journal Paper*. Furthermore, the model is cumbersome to read due to the large number of invisible tasks needed to express the flexible nature of the relations. Note that for Heuristics Miner it is possible to use other configurations, e.g., one with a lower dependency threshold. This would result in a model that better captures the behavior in the event log, but this solution would include a very generic model in which every transition can be executed in any order.

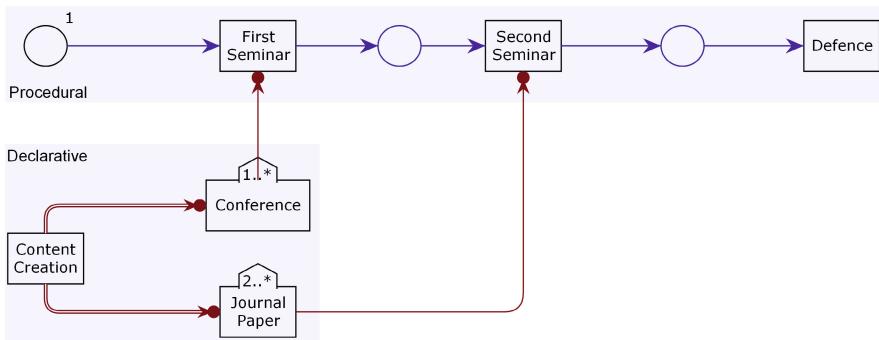


Fig. 1. A multi-paradigm process model representing a PhD student’s progress flow. The model contains procedural behavior in the form of Petri net fragments, supplemented with Declare constraints.

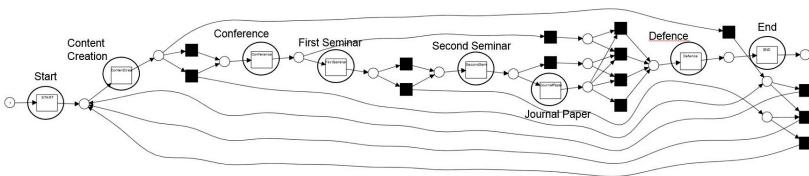


Fig. 2. A Petri net retrieved from the mined causal net produced by Flexible Heuristics Miner (default settings)

4 Advantages of a Combined Mining Approach

When comparing approaches for both paradigms, one of the main differences can be found in the granularity of capturing relations between activities. While procedural miners’ outcomes rely mainly on rather local information, declarative constraints capture rules supported on trace level, which can be interpreted as rather global information of the process. Both types of results carry different control flow knowledge, which can be combined to gain different insights. Consider this simple example: trace (a, b, c, a, a, b) fulfills both a *Succession* (a, b) and a local direct succession constraint ($a \rightarrow b$ or $a > b$, as often used in procedural process mining algorithms such as Heuristics Miner) between a and b , while (b, c, a, b, c, a, a, b) only fulfills the local constraint. Either the constraint is too restrictive for (the log which contains) this trace, or the model can be pruned by using the *Succession* template as counter evidence for the direct succession. A procedural miner would include the relation between a and b as $a \rightarrow b$, while a declarative miner would derive a *Response* constraint which lacks the *Precedence* part of a *Succession*. As such, it is better capable of deriving different types of relations, and is more expressive in this respect.

The main benefits of mining a multi-paradigm process model can be summarized as follows. On the one hand, a procedural model can benefit from the

addition of declarative constraints in order to uncover relations between activities that previously remained hidden. In this sense, the declarative constraints enhance the model which can better fit the parts of the log that were previously hard to capture. Since rules are defined over the full execution path, they are also better suited to represent, amongst others, duplicate tasks and long-distance dependencies. Furthermore, flexible parts of a log that are not captured (well) by procedural models (as they remained either too restrictive or too general) can be represented with declarative constraints to retrieve them in a more correct and readable way. Although capturing flexible behavior might be possible with procedural models, the sequential information would end up in a very convoluted and unstructured graph of loops, splits and joins, and arrows pointing every direction due to the ad-hoc appearance of activities as can be seen in Figure 2. Since most Declare rules represent behavior that can be labeled as non-trivial token games, they are better able to retrieve such parts of an event log. For example, expressing *Alternate Precedence* in a Petri net is a challenging task, leading to the usage of artificial model constructs to approximate the same state space.

On the other hand, declarative process models can benefit from the structuring and representation that procedural model discoverers offer, thus making represented flows more readable and more defined where no flexibility is needed, i.e. for a very fixed process sequence. In order to address these synergies between both paradigms, our approach starts from two sets of activities. The activities A in log L are divided in sets R and P , the declarative and procedural activities respectively. Note that $A = R \cup P$. By searching for Declare rules and procedural relations between both sets, the algorithm proposed in Section 5 captures the information that resides in the log by using both mining approaches.

5 Multi-paradigm Miner

The Multi-Paradigm Miner implementation² is based on the combination of Flexible Heuristics Miner (the Causal Net Retrieval implementation) and Declare Miner. Starting from the dependency graph, it identifies activities that are connected to a relatively higher number of other activities in the graph, as these can be considered a causal factor of the increase in potential process behavior. As such, we target them for inclusion in the set of activities that are subject to Declare mining in the second part of the discovery process. Finally, the mining result is displayed in a model which contains all behavior, which can be pruned optionally.

The Algorithm. By analyzing the strength of the direct succession metric (used in Heuristics Miner, which employs a certain threshold d called Dependency threshold for this purpose) between activities, one can retrieve the activities most closely related in a small window. Activities that have a lot of other

² The implementation and high resolution figures can be found at j.processmining.be/multiparadigmminer.

activities connected or are somewhat but not strongly connected, are candidates to be placed in the set of declarative activities $R \subseteq A$. Others that have few but strong connections to neighboring activities, are candidates to remain in the procedural basis of the model $P \subseteq A$. Phrased differently, we target activities with unclear direct succession relations, which can be an indication of the ad-hoc all-over-the-place occurrence of this activity, which results in non-structured and cluttered up sequential process models. Note that this approach also often captures the activities that cannot be fitted into the model and thus puts tasks that are connected only when the “All activities connected” option is chosen in Heuristics Miner in R .

To check for such activities we propose a metric called Activity Entropy (AE) which captures the average of the direct succession (DS) metrics between an activity and the others in the log where the dependency threshold d is not met. In other words, it captures weak dependencies. Procedural activities in a log will have a very low activity entropy, as most of the connections will be either strong ($> d$) or non-existing (close or equal to zero). Based on a given threshold e which is an input parameter of Multi-Paradigm Miner, a proportion of the log is withheld. The different values AE_i are ranked and $\lfloor |L|(1 - e) \rfloor$ activities are kept in the sorted set E . Furthermore, if there is a gap of $1/e$ between the values for AE of two activities in E , the activities ranked below the gap are removed. This procedure is established to avoid introducing too many activities in R and as a consequence possibly too many rules between them. Note, however, that a fully declarative model can be obtained by using 1 for e .

6 Evaluation

In this section, the results of our experiments are presented as to evaluate the capabilities of our approach. The following representation is used in the mixed Declare and Causal Graph figures:

- The full (blue) arcs represent the procedural behavior as introduced by Heuristics Miner. They form the Causal net of the model.
- The checkered activities exceed the entropy threshold.
- The dark activities (filled in red) fulfill the *Exactly1* constraint.
- The light activities (filled in gray) fulfill the *Existence* constraints.
- The striped arrows represent ConDec constraints, which are labelled.

In short, all checkered model constructs are the outcome of the Declare mining, while the full lines and activity borders are part of the Causal net.

The PhD Process. Figures 3 and 4 show the discovered multi-paradigm models in ProM. Even for a small entropy value (Figure 3), the activity *Content Creation* becomes subject to Declare constraint mining. By its constant enabledness it can appear anywhere in the workflow and clutter up a sequential process. By retrieving a few rules for the activity, we are able to represent it in a sense-making way in a mixed model. The constraints for single activities are

always applied, as they can only improve the understanding of the model. Since we use a simulated example, we apply a rule support of 100%. The model is already capable of capturing the initial model more correctly, as the relationships between *ContentCreation* and the other activities are correct. The arc between *SecondSeminar* and *JournalPaper* is still incorrect.

By raising the entropy level (Figure 4), more activities are added to the declarative set R , in this case *Conference* and *Journal Paper*. This makes sense given the model. Only constrained by the appearance of *Content Creation*, these activities are also rather unpredictable. Note that the procedural and Petri net part of the model is becoming smaller and smaller, while the Declare constraints offer the same behavior and more. Hence, a trade-off between precision and generalization exists.

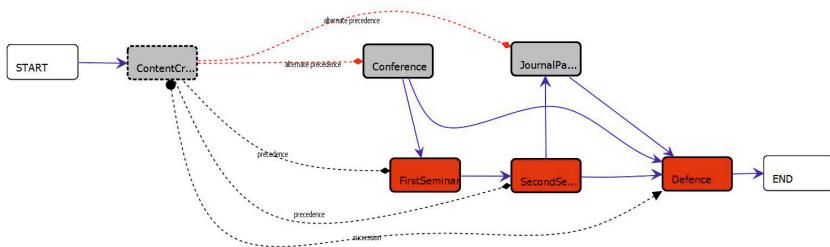


Fig. 3. Result of Multi-Paradigm Miner with $e = 0.2$

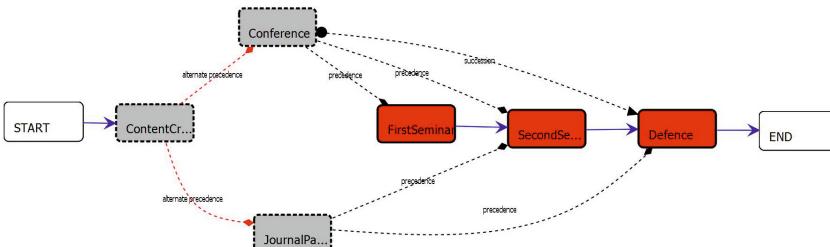


Fig. 4. Result of Multi-Paradigm Miner with $e = 0.5$

7 Conclusions and Future Work

This paper presents Multi-Paradigm miner, a process discovery technique that is capable of combining procedural and declarative process constructs into one discovery result. Hence, structured and more flexible parts of a workflow can be represented with the most appropriate constructs, thus allowing to find a trade-off between recall, precision, generalization, and simplicity in a more versatile

way. Hereto, we introduce the notion of a mixed-paradigm model. Initial experimental evaluations show that our technique can indeed discovery multi-paradigm models that are better and more useful than single-paradigm models.

However, to fully assess the power of the technique, the notion of precision, fitness and generalization needs to be introduced in future work. For both paradigms there already exist techniques such as [10,11] to evaluate fitness, which can serve as a starting point. An initial approach could include checking traces for Declare violations as proposed by [10] and use the move on model outcome to replay the corrected traces on, e.g., Petri nets.

Acknowledgments. This research is funded by FWO (Fonds voor Wetenschappelijk Onderzoek) Project G0804 13N.

References

1. van der Aalst, W.M.P.: *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Heidelberg (2011)
2. Maggi, F.M., Bose, R.P.J.C., van der Aalst, W.M.P.: Efficient discovery of understandable declarative process models from event logs. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 270–285. Springer, Heidelberg (2012)
3. Murata, T.: Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4), 541–580 (1989)
4. 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)
5. Pesic, M., Schonenberg, H., van der Aalst, W.M.: Declare: Full support for loosely-structured processes. In: 11th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2007, pp. 287–287. IEEE (2007)
6. Di Ciccio, C., Mecella, M.: A two-step fast algorithm for the automated discovery of declarative workflows. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 135–142. IEEE (2013)
7. Westergaard, M., Stahl, C., Reijers, H.A.: Unconstrainedminer: Efficient discovery of generalized declarative process models
8. Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Exploiting inductive logic programming techniques for declarative process mining. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 278–295. Springer, Heidelberg (2009)
9. Maggi, F.M., Slaats, T., Reijers, H.A.: The automated discovery of hybrid processes. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 392–399. Springer, Heidelberg (2014)
10. de Leoni, M., Maggi, F.M., van der Aalst, W.M.: An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. *Information Systems* (2014)
11. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.: Conformance checking using cost-based fitness analysis. In: 2011 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 55–64. IEEE (2011)

ODBASE 2014 PC Co-chairs Message

We are delighted to present the proceedings of the 13th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE) which was held in Amantea (Italy) 27-31 October 2014. The ODBASE Conference series provides a forum for research and practitioners on the use of ontologies and data semantics in novel applications, and continues to draw a highly diverse body of researchers and practitioners. ODBASE is part of the OnTheMove (OTM 2014) federated event composed of three interrelated yet complementary scientific conferences that together attempt to span a relevant range of the advanced research on, and cutting-edge development and application of, information handling and systems in the wider current context of ubiquitous distributed computing. The other two co-located conferences are CoopIS'14 (Cooperative Information Systems) and C&TC'14 (Cloud and Trusted Computing '14). Of particular relevance to ODBASE 2014 are papers that bridge traditional boundaries between disciplines such as databases, social networks, mobile systems, artificial intelligence, information retrieval, and computational linguistics.

This year, we received 47 paper submissions and had a program committee of 40 dedicated colleagues, including researchers and practitioners from diverse research areas. Special arrangements were made during the review process to ensure that each paper was reviewed by 3-4 members of different research areas. The result of this effort is the selection of high quality papers: fifteen regular papers (32%), ten short papers (21%), and four posters (9%). Their themes included studies and solutions to a number of modern challenges such as querying and management of linked data and RDF documents, ontology engineering, semantic matching and mapping, social network analysis, semantic web services, and streaming data. The scientific program is complemented with a very interesting keynote speech by Domenico Saccà on Mining and Posting Big Data on the Web.

We would like to thank all the members of the Program Committee for their hard work in selecting the papers and for helping to make this conference a success. We would also like to thank all the researchers who submitted their work. Last but not least, special thanks go to the members of the OTM team for their support and guidance.

We hope that you enjoy ODBASE 2014 and have a wonderful time in Amantea!

September 2014

Alfredo Cuzzocrea
Timos Sellis

Fuzzy XPath Queries in XQuery^{*}

Jesús M. Almendros-Jiménez¹, Alejandro Luna², and Ginés Moreno²

¹ Dep. of Informatics, University of Almería, Spain
`jalmen@ual.es`

² Dep. of Computing Systems, University of Castilla-La Mancha, Spain
`Gines.Moreno@uclm.es, Alejandro.Luna@alu.uclm.es`

Abstract. We have recently designed a fuzzy extension of the XPath language which provides ranked answers to flexible queries taking profit of fuzzy variants of *and*, *or* and *avg* operators for XPath conditions, as well as two structural constraints, called *down* and *deep*, for which a certain degree of relevance is associated. In this work, we describe how to implement the proposed fuzzy XPath with the XQuery language. Basically, we have defined an XQuery library able to fuzzily handle XPath expressions in such a way that our proposed fuzzy XPath can be encoded as XQuery expressions. The advantages of our approach is that any XQuery processor can handle a fuzzy version of XPath by using the library we have implemented.

Keywords: Fuzzy XPath, XML, XQuery.

1 Introduction

The *XPath* language [6] has been proposed as standard for XML querying and it is based on the description of the path in the XML tree to be retrieved. XPath allows to specify the name of nodes (i.e., tags) and attributes to be present in the XML tree together with Boolean conditions about the content of nodes and attributes. XPath querying mechanism is based on a Boolean logic: the nodes retrieved from an XPath expression are those matching the path of the XML tree, according to Boolean conditions.

Information retrieval requires the design of query languages able to adapt to user's preferences and providing ranked sets of answers. The *degree of satisfaction* of the user with respect to an answer can be measured in several ways. XPath lacks on mechanisms for giving priority to queries and ranking answers. In an XPath-based query, the main criteria to provide a certain degree of satisfaction are the *hierarchical deepness* and *document order*. Moreover, conditions on XPath expressions are usually of varying importance for a user, that is, the

* This work was supported by the EU (FEDER), and the Spanish MINECO Ministry (*Ministerio de Economía y Competitividad*) under grants TIN2013-45732-C4-2-P and TIN2013-44742-C4-4-R, as well as by the Andalusian Regional Government (Spain) under Project P10-TIC-6114.

user gives a *higher degree of importance to certain requirements* when satisfying his (her) wishes.

With this aim we have recently designed a fuzzy extension of XPath whose main aim is to provide mechanisms to assign priority to queries and to rank answers. Priorities are given by using fuzzy extensions of Boolean operators, while rankings are defined with regard to the location of a tag in the XML tree.

Firstly, we have proposed the incorporation to XPath of two structural constraints called *down* and *deep* for which a certain degree of relevance can be associated. So, whereas *down* provides a ranked set of answers depending on the path they are found from “top to down” in the XML document, *deep* provides a ranked set of answers depending on the path they are found from “left to right” in the XML document. Both structural constraints can be used together, assigning degree of importance with respect to the distance to the root XML element.

Secondly, we provide fuzzy variants of *and* and *or* for XPath conditions. We have enriched the arsenal of operators of XPath with fuzzy variants of *and* and *or*. Particularly, we have considered three versions of *and*: *and+*, *and*, *and-* (and the same for *or* : *or+*, *or*, *or-*) which make more flexible the composition of fuzzy conditions. Three versions for each operator which are obtained from our adaptation of the *Product*, *Lukasiewicz* and *Gödel* logics to the XPath paradigm. We claim that in our work the fuzzy versions provide a mechanism to force (and debilitate) conditions in the sense that stronger (and weaker) user preferences can be modeled with the use of stronger (and weaker) fuzzy conditions. The combination of fuzzy operators in queries permits to specify a ranked set of fuzzy conditions according to user’s requirements.

Finally, we have equipped XPath with an additional operator that is also traditional in fuzzy logic: the average operator *avg*. This operator offers the possibility to explicitly give weight to fuzzy conditions. Rating conditions by *avg*, solutions increase their weight in a proportional way. However, from the point of view of the user’s preferences, it forces the user to quantify his (her) wishes which, in some occasions, can be difficult to measure. For this reason, fuzzy versions of *and* and *or* are better choices in some circumstances.

In this work, we describe how to implement our fuzzy variant of the XPath language within the XQuery language. Basically, we have defined an XQuery library able to fuzzily handle XPath expressions in such a way that our proposed fuzzy XPath can be encoded as XQuery expressions.

The implementation of our fuzzy extension of XPath is based on an XQuery library of functions including *deep* and *down* operators, as well as the fuzzy operators *and+*, *and-*, *and*, *or+*, *or-*, *or* and *avg*. Using this library the user can replace Boolean operators by fuzzy versions in XPath expressions, as well as he (she) can call to *deep* and *down* operators, in order to obtain ranked sets of answers. The answers are shown with a *Retrieval State Value (RSV)* representing the degree of satisfaction. The answers can also be ordered with respect to the RSV making use of *descending* XQuery expression, as well as filtered with regard to a *threshold*.

The input documents in our proposal are *crisp XML documents*, but the answers to a query offer fuzzy information, that is, a *RSV for each answer*. Therefore our approach is focused on the handling of standard XML documents, in which the user can retrieve information, ranked by a certain degree of satisfaction. We have decided to implement fuzzy XPath within XQuery by providing an XQuery library of fuzzy operators. It makes possible that our library can be used from any XQuery processor to query any XML document with crisp information.

Although the input of a query is a crisp XML document, the library assigns internally and, in a transparent way to the user, a RSV to each of node of interest in the document. The RSVs assigned to each node of interest are used to compute the RSV of the answer. It makes the implementation a non-trivial task. Starting from a crisp XML document as input, our implementation annotates at run-time a RSV to each node of the query result. It also involves to dynamically annotate RSVs of nodes in subqueries. Additionally, *where* and *return* expressions of XQuery become XQuery functions in order to handle fuzzy conditions and RSVs, respectively.

Finally, let us remark that we have previously developed in [1,2,3,4,5] an implementation¹ of our fuzzy XPath using the *FLOPER “Fuzzy LLogic Programming Environment for Research”* tool² which is based on Multi-Adjoint Logic Programming (MALP) [16,17]. There we made use of the fuzzy logic nature of FLOPER to implement fuzzy XPath by using fuzzy logic rules. Here the implementation has to adapt a Boolean logic based language (i.e., XQuery) to obtain the same behavior as in MALP. The implementation in XQuery can be downloaded from <http://dectau.uclm.es/fuzzyXPath/>.

The structure of the paper is as follows. Section 2 will describe the fuzzy version of XPath and some examples. Section 3 will present the implementation in XQuery. Section 4 will show the same of examples than Section 2, written in XQuery. Section 5 will present related work. Finally, Section 6 will conclude and present future work.

2 A Flexible XPath Language

Our proposal of fuzzy XPath is defined by the grammar of Figure 1. Basically, the extension of XPath is as follows:

- A given XPath expression can be adorned with «[DEEP = r_1 ; DOWN = r_2]» which means that the *deepness* of elements is penalized by r_1 and that the *order* of elements is penalized by r_2 , and such penalization is proportional to the distance (i.e., the length of the branch and the weight of the tree, respectively). In particular, «[DEEP = 1; DOWN = r_2]» can be used for penalizing only w.r.t. document order. *deep* works for `//`, that is, the deepness in the XML tree is only computed when descendant nodes are explored, while

¹ <http://dectau.uclm.es/fuzzyXPath/>

² <http://dectau.uclm.es/floper>.

```

xpath := ['[`deep-down']' ]path
path := literal | text() | node | @att | node/path | node//path
node := QName | QName[cond]
cond := xpath op xpath | xpath num-op number
deep := DEEP=number
down := DOWN=number
deep-down := deep | down | deep ';' down
num-op := > | = | < | <>
fuzzy-op := and | and+ | and- | or | or+ | or- |
           avg | avg{number,number}
op := num-op | fuzzy-op

```

Fig. 1. Fuzzy XPath Grammar

$$\begin{array}{lll}
& \&_P(x,y) = x * y & |_P(x,y) = x + y - x * y \quad \text{Product: and/or} \\
& \&_G(x,y) = \min(x,y) & |_G(x,y) = \max(x,y) \quad \text{Gödel: and+/or-} \\
& \&_L(x,y) = \max(x+y-1,0) & |_L(x,y) = \min(x+y,1) \quad \text{Łukasiewicz: and-/or+}
\end{array}$$

Fig. 2. Fuzzy Logical Operators

down works for both */* and *//*. Let us remark that *deep* and *down* can be used anywhere, and many times in an XPath expression.

- We consider three versions for each one of the conjunction and disjunction operators (also called connectives or aggregators) which are based in the so-called *Product*, *Gödel* and *Łukasiewicz* fuzzy logics. The *Gödel* and *Łukasiewicz* logic based fuzzy symbols³ are represented in our application by *and+*, *and-*, *or-* and *or+*, in contrast with product logic operators *and* and *or* (see Figure 2). Adjectives like *pessimistic*, *realistic* and *optimistic* are sometimes applied to the *Łukasiewicz*, *Product* and *Gödel* fuzzy logics since operators satisfy that, for any pair of real numbers x and y in $[0, 1]$: $0 \leq \&_L(x,y) \leq \&_P(x,y) \leq \&_G(x,y) \leq 1$ and the contrary for the disjunction operations: $0 \leq |_G(x,y) \leq |_P(x,y) \leq |_L(x,y) \leq 1$. So, note that it is more difficult to satisfy a condition based on a pessimistic conjunctive/disjunctive (i.e., *and-/or-* inspired by the *Łukasiewicz* and *Gödel* logics, respectively) than with *Product* logic based operators (i.e., *and/or*), while the optimistic versions of such connectives (i.e., *and+/or+*) are less restrictive, obtaining a greater set of answers. This is a consequence of the following chain of inequalities: $0 \leq \text{and-}(x,y) \leq \text{and}(x,y) \leq \text{and+}(x,y) \leq \text{or-}(x,y) \leq \text{or}(x,y) \leq \text{or+}(x,y) \leq 1$. Therefore users should refine queries by choosing operators in the previous sequence from left to right (or from right to left), till finding solutions satisfying in a stronger (or weaker) way the requirements.
- Finally, the *avg* operator is defined too in a *weighted* way. Assuming two given RSVs r_1 and r_2 , *avg* is defined as $(r_1 + r_2)/2$, and *avg*{ a, b } is defined as $(a * r_1 + b * r_2)/a + b$.

³ The fuzzy logic community frequently uses the terms *t-norm* and *t-conorm* for expressing generalized versions of conjunctions and disjunctions.

```

<hotels>
<hotel name="Melia">
  <close_to>Gran Via
    <close_to>Callao</close_to>
    <close_to>Plaza de Espana</close_to>
  </close_to>
  <services>
    <pool></pool>
    <metro>150</metro>
  </services>
  <price>100</price>
</hotel>
<hotel name="NH">
  <close_to>Sol
    <close_to>Gran Via</close_to>
    <close_to>Callao</close_to>
  </close_to>
  <services>
    <metro>300</metro>
  </services>
  <price>150</price>
</hotel>
<hotel name="Hilton">
  <close_to>Moncloa
    <close_to>Gran Via</close_to>
    <close_to>Sol</close_to>
  </close_to>
  <services>
    <metro>150</metro>
  </services>
  <price>50</price>
</hotel>
<hotel name="Tryp">
  <close_to>Cibeles
    <close_to>Alcala
      <close_to>Gran Via</close_to>
    </close_to>
    <close_to>Retiro</close_to>
  </close_to>
  <services>
    <pool></pool>
    <metro>10</metro>
  </services>
  <price>575</price>
</hotel>
<hotel name="Sheraton">
  <close_to>Recoletos
    <close_to>Cibeles</close_to>
    <close_to>Gran Via
      <close_to>Sol</close_to>
    </close_to>
  </close_to>
  <close_to>Sol</close_to>
  <services>
    <pool></pool>
    <metro>300</metro>
  </services>
  <price>475</price>
</hotel>
</hotels>

```

Fig. 3. Input XML document collecting Hotel's information

In general, an extended XPath expression defines, w.r.t. an XML document, a sequence of subtrees of the XML document where each subtree has an associated RSV. XPath conditions, which are defined as fuzzy operators applied to XPath expressions, compute a new RSV from the RSVs of the involved XPath expressions, which at the same time, provides a RSV to the node.

2.1 Examples of Fuzzy XPath

In order to illustrate the language, let us see some examples of flexible queries in XPath. We will take as input document the one shown in Figure 3. The example shows a sequence of hotels where each one is described by *name* and

price, proximity to streets (*close_to*⁴) and provided services (*pool* and *metro*-together with distance-). In the example, we assume that *document order* has the following semantics⁵. The tag *close_to* specifies the proximity to a given street. However, the order of *close_to* tags is relevant, and the top streets are closer than the streets at the bottom. In other words, the case:

```
hotel_H
  close_to street_A
  close_to street_B
```

implies that hotel H is near to both streets A and B, but closer to A than to B. The nesting of *close_to* has also a relevant meaning. While a given street A can be close to the hotel H, the streets close to A are not necessarily close to the hotel H. In other words, in the case:

```
hotel_H
  close_to street_A
    close_to street_B
```

the street B is near to street A, and street A is close to hotel H, which implies that street B is also close to hotel H, but no so close as street A. For instance, H can be situated at the end of street A, and B can cross A at the beginning. We can say, in this case, that B is an *adjacent* street to H, while A is *close* to H. This means that when looking for a hotel close to a given street, the highest priority should be assigned to streets close to the hotel, while adjacent streets should be relegated to lower priority. Particularly, when the user tries to find hotels very close to a given street a high *down* value and a low *deep* value should be provided, whereas in the case the user tries to find hotels in the neighborhood of a street, a high *deep* and low *down* should be requested.

2.2 Examples of DEEP and DOWN

In our first example, we focus on the use of *down*. Let us now suppose that the user is interested to find a hotel close to Sol street. This might be his (her) first tentative looking for a hotel. Using crisp XPath he (she) would formulate:

```
<< /hotels/hotel[close_to/text() = "Sol"]/@name >>
```

However, it gives the user the set of hotels close to Sol without distinguishing the degree of proximity. The fuzzy version of XPath permits to specify a degradation of answers, in such a way that the user reformulates the query as:

```
<< /hotels/hotel[[DOWN = 0.9]close_to/text() = "Sol"]/@name >>
```

The query specifies that *close_to* tag is degraded by 0.9 from top to down. In other words, when Sol is found close to a hotel, the position in which it occurs gives a different satisfaction value. In this case, we will obtain:

⁴ Let us remark that *close_to* is not a fuzzy relation in our approach. As was commented before, input XML documents are crisp.

⁵ The document order semantics can vary from one document to another. This example has been chosen to show the expressive power of our query language. Another kind of document can have a different order semantics and therefore the queries should be adapted.

```
<result>
  <result rsv="1.0">NH</result>
  <result rsv="0.9">Sheraton</result>
</result>
```

Fortunately, we have found a hotel (NH) which is very close to Sol, and one (Sheraton) which is a little bit farther from Sol. Let us remark the previous example and the other examples of the Section show the results in order of satisfaction degree.

Let us now suppose that we are looking for a hotel close to Callao. In this case, we can try to make the same question:

```
<< /hotels/hotel[[DOWN = 0.9]close_to/text() = "Callao"]/@name >>
```

However, the result is empty. Therefore we can try to relax the query by changing ‘/’ by ‘//’:

```
<< /hotels/hotel[[DOWN = 0.9]//close_to/text() = "Callao"]/@name >>
```

Now, we will find answers, however, we will not be able to distinguish the proximity of the hotels. Our fuzzy version of XPath permits to specify how the solutions are degraded but not only taking into account the order but also the deepness. In other words, there would be useful to give different weights to be a close street, and to be an adjacent street. Therefore we can use the query:

```
<< /hotels/hotel[[DEEP = 0.5; DOWN = 0.9]//close_to/text() = "Callao"]/@name >>
```

obtaining the following results:

```
<result>
  <result rsv="0.5">Melia</result>
  <result rsv="0.45">NH</result>
</result>
```

Thus Melia is near to Callao, and NH is a little bit farther than Melia.

The use of *deep* combined with *down* could be considered as the best choice. However, *deep* can be used alone when the user only wants to penalize adjacency. Whenever we like to search hotels near to Gran Via street, degrading adjacent streets with a factor of 0.5, we can consider the following query (and we obtain the following result):

```
<< //hotel[[DEEP = 0.5]//close_to/text() = "Gran Via"]/@name >>
```

```
<result>
  <result rsv="1.0">Melia</result>
  <result rsv="0.5">NH</result>
  <result rsv="0.5">Hilton</result>
  <result rsv="0.5">Sheraton</result>
  <result rsv="0.25">Tryp</result>
</result>
```

We can see that *Melia* is close to *Gran Via*, while *NH*, *Hilton* and *Sheraton* are situated in adjacent streets of *Gran Via*. *Tryp* is the farthest hotel.

2.3 Examples of AVG

Let us now suppose that the user is interested in a hotel combining two services like pool and metro. Instead of using classical *and/or* connectives for mixing

both features, we can obtain more flexible estimations on *RSV* values by using the *avg* operator as follows:

```
<< //hotel[services/pool avg services.metro]/@name >>
```

thus obtaining the following results:

```
<result>
  <result rsv="1.0">Melia</result>
  <result rsv="1.0">Tryp</result>
  <result rsv="1.0">Sheraton</result>
  <result rsv="0.5">NH</result>
  <result rsv="0.5">Hilton</result>
</result>
```

By using the *avg* fuzzy operator, the user finds that *Melia*, *Tryp* and *Sheraton* have pool and metro, while *NH* and *Hilton* lack on one of them.

Let us now suppose that the importance of the metro is the double of the importance of the pool. In this case, the user can formulate the query as follows:

```
<< //hotel[pool avg{1,2} services.metro]/@name >>
```

obtaining the following results:

```
<result>
  <result rsv="1.0">Melia</result>
  <result rsv="1.0">Tryp</result>
  <result rsv="1.0">Sheraton</result>
  <result rsv="0.666667">NH</result>
  <result rsv="0.666667">Hilton</result>
</result>
```

We can see in the results that *NH* and *Hilton* increase the degree of satisfaction w.r.t. the previous query given that they have metro station.

Let us now suppose the user is looking now for hotels giving more importance to the fact that the price of the hotel is lower than 150 euros than to the proximity to Sol street. The user can formulate the query as follows, obtaining the results below:

```
<< //hotel[[DEEP = 0.8]//close_to/text() = "Sol" avg{1,2} //price/text() < 150]/@name >>
```

```
<result>
  <result rsv="0.933333">Hilton</result>
  <result rsv="0.666667">Melia</result>
  <result rsv="0.333333">NH</result>
  <result rsv="0.333333">Sheraton</result>
</result>
```

2.4 Examples of AND

In the following queries we express the following requirement: hotels near to Gran Via, near to a metro station, having pool, with greater preference (3 to 2) to pool than metro. We will use *and+*, *and* and *and-* which provide different levels of exigency, which are demonstrated in the results.

```
<< //hotel[([DEEP = 0.5]//close_to/text() = "GranVia") and+(//pool avg{3,2} //metro/text() < 200)]/@name >>
```

```
<result>
  <result rsv="1.0">Melia</result>
  <result rsv="0.5">Sheraton</result>
  <result rsv="0.4">Hilton</result>
  <result rsv="0.25">Tryp</result>
</result>
```

```
<< //hotel[([DEEP = 0.5]//close_to/text() = "GranVia") and(//pool avg{3,2} //metro/text() < 200)]/@name >>
```

```
<result>
  <result rsv="1.0">Melia</result>
  <result rsv="0.3">Sheraton</result>
  <result rsv="0.25">Tryp</result>
  <result rsv="0.2">Hilton</result>
</result>
```

```
<< //hotel[([DEEP = 0.5]//close_to/text() = "GranVia") and - (//pool avg{3,2} //metro/text() < 200)]/@name >>
```

```
<result>
  <result rsv="1.0">Melia</result>
  <result rsv="0.25">Tryp</result>
  <result rsv="0.1">Sheraton</result>
</result>
```

So, in the first case (the least demanding and optimistic) we obtain four hotels (Melia, Sheraton, Hilton and Tryp), as well as in the second case (a little bit more exigent) while third table (the strongest one) lists three candidates (Melia, Tryp and Sheraton). Sheraton and Hilton are degraded using *and* and *and-*.

3 XQuery Library for Fuzzy XPath

We can summarize the elements of the implementation as follows:

3.1 Elements of the Library

1. The *deep* and *down* operators become XQuery functions that take as arguments a context node, an XPath expression and the value (a real number in $[0,1]$) assigned to deep and down, respectively. For combining deep and down an XQuery function is defined having as argument two real values in $[0,1]$:

```
declare function f:deep($node,$xpath,$deep)
declare function f:down($node,$xpath,$down)
declare function f:deep_down($node,$xpath,$deep,$down)
```

2. Fuzzy versions of Boolean operators *and*, *or* have been defined as XQuery functions, each one for each fuzzy logic we have considered (i.e., *Product*, *Lukasiewicz* and *Gödel*):

```
declare function f:andP($left,$right)
declare function f:orP($left,$right)
declare function f:andG($left,$right)
declare function f:orG($left,$right)
declare function f:andL($left,$right)
declare function f:orL($left,$right)
```

3. Operators *avg* and *avg{a,b}* have been defined as XQuery functions:

```
declare function f:avg($left,$right)
declare function f:avg_ab($left,$right,$a,$b)
```

4. Fuzzy versions of XQuery expressions *where* and *return* have been defined.

In order to make transparent to the user the incorporation of RSVs, we have defined a new version of the *return* expression, called *returnF*, which transparently carries out the computation of the RSVs of the answers.

Similarly, since XQuery works with a Boolean logic, the introduction of fuzzy versions of the operators, forces us to define a new version of the *where* expression, called *whereF*, which transparently carries out the computation of the RSVs from fuzzy conditions. *ReturnF* has as parameters the context node and an XPath expression. *WhereF* has as parameters the context node and a fuzzy condition.

```
declare function f:whereF($node,$fuzzycond)
declare function f:returnF($node,$xpath)
```

5. Fuzzy versions of comparison operators for XPath expressions have been defined as XQuery functions. Similarly to *whereF*, comparison operators have been adapted to handle the RSVs:

```
declare function f:equalF($left,$right)
declare function f:lessF($left,$right)
declare function f:greaterF($left,$right)
```

3.2 Implementation of the Library

In order to implement our library in XQuery we have used the *XQuery Module* available in the *BaseX* processor [13]. In particular, we make use of the function *eval* that makes possible the manipulation of XPath expressions. This function is also available for *Exist* [15] and *Saxon* [14] processors. For instance, *down* is defined as follows:

```
declare function f:down($nodes,$query, $down){
  let $docDown := document{f:down_aux($nodes/*,$down,(),())}
  let $docQ := xquery:eval(concat('$', $query), map { '$x' := $docDown})
  let $docL := xquery:eval(concat('$', $query), map { '$x' := $nodes})
  return f:putListRSV($docL,f:getListRSV($docQ))
};
```

deep is defined as follows:

```
declare function f:deep($doc as node()*, $query, $deep as xs:double){
  let $docDeep := document{f:deep_aux($doc/*,$deep,1)}
  let $docQ := xquery:eval(concat('$', $query), map { '$x' := $docDeep})
  let $docL := xquery:eval(concat('$', $query), map { '$x' := $doc})
  return f:putListRSV($docL,f:getListRSV($docQ))
};
```

and *deep_down* is defined as follows:

```

declare function f:deep_down($nodes as node(), $query, $deep as xs:double,
    $down as xs:double){
    let $docDown := document{f:down_aux($nodes/*,$down,(),())}
    let $docDeep := document{f:deep_aux($docDown/*,$deep,1)}
    let $docQ := xquery:eval(concat(''$x'', $query), map { ''$x'' := $docDeep})
    let $docL := xquery:eval(concat(''$x'', $query), map { ''$x'' := $nodes})
    return f:putListRSV($docL,f:getListRSV($docQ))
};
```

Each fuzzy operator has been defined as a function, for instance, *and* (*Product* logic), *or+* (*Gödel* logic), *avg*, and *avg{a,b}* are defined as follows:

```

declare function f:andP($cond1,$cond2)
{
    let $tv1 := f:truthValue($cond1)
    let $tv2 := f:truthValue($cond2)
    return $tv1*$tv2
};

declare function f:orG($cond1,$cond2)
{
    let $tv1 := f:truthValue($cond1)
    let $tv2 := f:truthValue($cond2)
    return
        if ($tv1 > $tv2) then $tv1
        else $tv2
};

declare function f:avg($cond1,$cond2)
{
    let $tv1 := f:truthValue($cond1)
    let $tv2 := f:truthValue($cond2)
    return (xs:double($tv1)+xs:double($tv2)) div (2)
};

declare function f:avg_ab($cond1,$cond2, $a, $b)
{
    let $tv1 := f:truthValue($cond1)
    let $tv2 := f:truthValue($cond2)
    return (xs:double($tv1)*$a+xs:double($tv2)*$b) div ($a+$b)
};
```

4 Examples of Fuzzy XPath in XQuery

Now, we show how the fuzzy XPath queries of Section 2 can be written in XQuery.

4.1 Examples of DEEP and DOWN

Let us now suppose the following fuzzy XPath query:

```
<< /hotels/hotel[[DOWN = 0.9]close_to/text() = "Sol"]/@name >>
```

We can now write the same query in XQuery as follows:

```

for $x in doc('hotels.xml')/hotels/hotel
let $y := f:whereF($x,f:equalF(f:down($x,'/close_to',0.9),'Sol'))
let $z := f:returnF($y,'@name')
order by $y/@rsv descending
return $z
```

We can see that fuzzy XPath expressions are written as XQuery expressions. This is the same kind of transformation from crisp XPath to XQuery. For instance:

`<< /hotels/hotel[close_to/text() = "Sol"]/@name >>`

can be translated into:

```
for $x in doc("hotels.xml")/hotels/hotel
where $x/close_to/text()="Sol"
return $x/@name
```

In the fuzzy case, “=” is transformed into *equalF*, and *where* as well as *return* become XQuery functions, with an extra argument to represent the context node. The query makes use of the function *down* of the library to compute the RSVs associated to *close_to*. In addition, the attribute *rsv*, which has been (internally) added to the output document, can be handled to show the answer in a sorted way, and even to define a threshold.

Let us now consider the following query, that uses *deep* and *down*:

`<< /hotels/hotel[[DEEP = 0.5; DOWN = 0.9]//close_to/text() = "Callao"]/@name >>`

We can now write the same query in XQuery using the function *deep_down*:

```
for $x in doc('hotels.xml')/hotels/hotel
let $y :=
  f:whereF($x, f:equalF(f:deep_down($x,'//close_to',0.5,0.9), 'Callao'))
let $z := f:returnF($y,'/ename')
order by $y/@rsv descending
return $z
```

4.2 Examples of AVG

Let us now suppose the following fuzzy XPath expression that makes use of the *avg* operator.

`<< //hotel[services/pool avg services.metro]/@name >>`

Here, we use the function *avg* of the library, having as parameters both sides of the fuzzy condition:

```
for $x in doc('hotels.xml')//hotel
let $y := f:whereF($x, f:avg($x/services/pool,$x/services/metro))
let $z := f:returnF($y,'/ename')
order by $y/@rsv descending
return $z
```

The same can be said for the following query, using *avg{a,b}* having as parameters *a* and *b*.

`<< //hotel[services/pool avg{1,2} services.metro]/@name >>`

```
for $x in doc('hotels.xml')//hotel
let $y := f:whereF($x,f:avg_ab($x/services/pool, $x/services/metro,1,2))
let $z := f:returnF($y,'/ename')
order by $y/@rsv descending
return $z
```

4.3 Examples of AND

Let us now suppose the following queries that combine *deep* and *avg*, and *deep* and *and+*, respectively:

<i>Query</i>	<i>16Kb</i>	<i>700Kb</i>	<i>4.8Mb</i>	<i>15.4Mb</i>
<i>Examined nodes in Q1</i>	28	148	298	448
<i>Examined nodes in Q2</i>	25	145	295	445
<i>Tree Depth</i>	21	101	201	301
Q1	7.09 ms	25.47 ms	123.66 ms	461.6 ms
down in Q1	12.52 ms	107.1 ms	481.24 ms	2853.36 ms
deep in Q1	10.08 ms	74.17 ms	510.74 ms	1953.31 ms
deep and down in Q1	69.97 ms	102.0 ms	685.87 ms	7315.59 ms
Q2	5.77 ms	57.96 ms	172.03 ms	529.18 ms
avg in Q2	36.59 ms	1266.99 ms	9729.49 ms	60426.28 ms

Fig. 4. Benchmarks

```
<< //hotel[[DEEP = 0.8]]//close_to/text() = "Sol" avg{1,2} //price/text() < 150]//@name >>
```

```
for $x in doc('hotels.xml')//hotel
let $y := f:whereF($x, f:avg_ab(f:equalF(f:deep($x,'//close_to',0.8),'Sol'),
                                    $x//price/text()<150,1,2))
let $z := f:returnF($y,'@name')
order by $y/@rsv descending
return $z
```

```
<< //hotel[[DEEP = 0.5]]//close_to/text() = "GranVia" and+(//pool avg{3,2} //metro/text() < 200)]//@name >>
```

```
for $x in doc('hotels.xml')//hotel
let $y := f:whereF($x,f:andG(f:equalF(f:deep($x,'//close_to',0.5),'GranVia'),
                                f:avg_ab($x//pool,$x//metro<200,3,2)))
let $z := f:returnF($y,'@name')
order by $y/@rsv descending
return $z
```

4.4 Benchmarks

Now, we would like to show the benchmarks we have obtained using our library. We have used the BaseX Query processor in a Intel Core 2 Duo 2.66 GHz Mac OS machine. We have tested our library using data sets of different sizes. We have used as data sets traces of execution of MALP programs developed under our FLOPER tool. The FLOPER tool generates traces in XML format, with a high degree of tag nesting when a recursive program is executed. These data sets facilitate the testing of our structural based operators *deep* and *down*.

In Figure 4 we can see the results, where we indicate the number of nodes examined in each query, as well as the depth of the tree. We have compared the execution times for two XPath expressions in crisp and fuzzy versions. The first query is Q1:

```
« //node/goal »
```

and the second query is Q2:

```
« //node[goal[contains(text(),"p(")] and
substitution[contains(text(),"g(")]]//goal »
```

5 Related Work

Fuzzy versions of XQuery have been previously studied in some works. The closest to our approach is [12], in which preferences can be described by queries in order to retrieve discriminated answers by user's preferences. FLOWR expressions are extended to cover with fuzzy values and answers. The main aim of their work is to extend XQuery with definition of fuzzy terms: good, cheap, high, young, etc., defined as fuzzy predicates that can be imposed in XPath expressions. They extend XQuery datatypes with *xs:truth* and incorporate *xml:truth* as attribute to represent degree of satisfaction. Nevertheless, they lack on an implementation, and therefore we cannot compare our proposal with them, although we believe that a similar technique we have proposed here can be used. In [18], they also extends the syntax of XQuery, in particular, the expression *where* to cover with priority and thresholding. Their approach is focused on querying fuzzy XML data, and therefore their proposal is different from our. They have developed an implementation using Java on top of the Exist [15] XQuery processor. A fuzzy query is transformed into standard XQuery to be executed. Fuzzy data querying is also the main aim of the work of [19], in which they propose a fuzzy XML Schema and algebraic operators to handle fuzzy data over an schema. They provide transformations from the algebraic operators to XQuery (and XPath) expressions. Again, their approach is different from our, since they work with fuzzy XML data as input.

Fuzzy versions of XPath have been previously studied in some works. The closest works to our proposal are [7,8] in which authors introduce in XPath flexible matching by means of fuzzy constraints called *close* and *similar* for node content, together with *below* and *near* for path structure. In addition, they have studied the *deep-similar* notion for tree matching, and fuzzy versions for *not*, *and* and *or* operators. In order to provide ranked answers they assign a *RSV* to each item. Our work is similar to the proposed by [7,8]. The *below* operator of [7,8] is equivalent to our proposed *down*: both extract elements that are direct descendants of the current node, and the penalization is proportional to the distance. The *near* operator of [7,8], which is defined as a generalization of *below*, ranks answers depending on the distance to the required node, in any XPath axis. Our proposed *deep* ranks answers depending of the distance to the current node, but the considered nodes can be direct and non direct descendants. Therefore our proposed *deep* combined with *down* is a particular case of *near*. To have the same expressive power as *near* we could incorporate to our framework a new operator to rank answers from bottom to up. With respect to *similar* and *close* operators proposed in [7,8], our framework lacks similarity relations and rather focuses on structural (i.e. path-based) flexibility.

In [10], the authors propose to give a satisfaction degree to XPath expressions based on associating weights to XPath steps. Relaxing XPath expressions when the path does not match the XML schema is the main goal of this work. They have studied how to compute the best *k* answers. In this line, in [9,11] XPath relaxation is studied given some rules for query rewriting: axis relaxation, step deletion and step cloning, among others. The proposed *deep-similar* notion of

[7,8] can be also considered a relaxation technique of XML tree equality. Our work has some similarities with these proposals: *deep* and *down*, and also the use of *avg* operator, are mechanisms for relaxing queries and giving priority to paths and answers. We have also studied in [3] how to introduce axis relaxation, step deletion and step cloning in our approach, but the proposed implementation does not still include these mechanisms. It is considered as future work.

6 Conclusions and Future Work

In this paper we have presented an XQuery based implementation of a fuzzy version of XPath. Fuzzy XPath incorporates mechanisms to rank answers depending on the location of the item in the XML tree of input, as well as to give priority to queries. The output of a query contains an RSV in each item according to the user's preferences. We have described the elements of the XQuery library that make possible to express fuzzy queries against crisp XML data. As future work, we plan the following steps. Firstly, to incorporate new mechanisms of searching and ranking to queries. We have previously studied [3,4] how to penalize answers when a given XPath expression is incorrect, and tags have to be jumped, switched and added. We believe that these mechanisms can be implemented also in XQuery. Secondly, we would like to extend our work to other fuzzy logic mechanism (vagueness, similarity, etc). Another direction we can take is to introduce the operators in XQuery and then use a parser to translate the user expression in standard XQuery language. Finally, we would like to improve the performance of our implementation, for instance, thanks to the use of thresholding, following [5]. Up to now, thresholding is achieved on the output of the query, and dynamic thresholding would improve the performance.

References

1. Almendros-Jiménez, J.M., Luna, A., Moreno, G.: Fuzzy Logic Programming for Implementing a Flexible XPath-based Query Language. *Electr. Notes Theor. Comput. Sci.* 282, 3–18 (2012)
2. Almendros-Jiménez, J.M., Luna, A., Moreno, G.: A Flexible XPath-based Query Language Implemented with Fuzzy Logic Programming. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 186–193. Springer, Heidelberg (2011)
3. Almendros-Jiménez, J.M., Luna, A., Moreno, G.: A XPath Debugger Based on Fuzzy Chance Degrees. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM-WS 2012. LNCS, vol. 7567, pp. 669–672. Springer, Heidelberg (2012)
4. Almendros-Jiménez, J.M., Luna, A., Moreno, G.: Annotating “Fuzzy Chance Degrees” when Debugging Xpath Queries. In: Rojas, I., Joya, G., Cabestany, J. (eds.) IWANN 2013, Part II. LNCS, vol. 7903, pp. 300–311. Springer, Heidelberg (2013)
5. Almendros-Jiménez, J.M., Luna, A., Moreno, G.: Dynamic Filtering of Ranked Answers When Evaluating Fuzzy XPath Queries. In: Cornelis, C., Kryszkiewicz, M., Ślęzak, D., Ruiz, E.M., Bello, R., Shang, L. (eds.) RSCTC 2014. LNCS, vol. 8536, pp. 319–330. Springer, Heidelberg (2014)

6. Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., Siméon, J.: XML path language (XPath) 2.0. W3C (2007)
7. Campi, A., Damiani, E., Guinea, S., Marrara, S., Pasi, G., Spoletini, P.: A fuzzy extension of the XPath query language. *Journal of Intelligent Information Systems* 33(3), 285–305 (2009)
8. Damiani, E., Marrara, S., Pasi, G.: FuzzyXPath: Using fuzzy logic and IR features to approximately query XML documents. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (eds.) IFSA 2007. LNCS (LNAI), vol. 4529, pp. 199–208. Springer, Heidelberg (2007)
9. Fazzinga, B., Flesca, S., Furfarò, F.: On the expressiveness of generalization rules for XPath query relaxation. In: Proceedings of the Fourteenth International Database Engineering & Applications Symposium, pp. 157–168. ACM (2010)
10. Fazzinga, B., Flesca, S., Pugliese, A.: Top- k Answers to Fuzzy XPath Queries. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 822–829. Springer, Heidelberg (2009)
11. Fazzinga, B., Flesca, S., Furfarò, F.: Xpath query relaxation through rewriting rules. *IEEE Transactions on Knowledge and Data Engineering* 23(10), 1583–1600 (2011)
12. Goncalves, M., Tineo, L.: Fuzzy XQuery. In: Ma, Z., Yan, L. (eds.) Soft Computing in XML Data Management. STUDFUZZ, vol. 255, pp. 133–163. Springer, Heidelberg (2010)
13. Grün, C.: BaseX. The XML Database (2014), <http://basex.org>
14. Kay, M.: Ten reasons why saxon xquery is fast. *IEEE Data Eng. Bull.* 31(4), 65–74 (2008)
15. Meier, W.: eXist: An open source native XML database. In: Chaudhri, A.B., Jeckle, M., Rahm, E., Unland, R. (eds.) NODE-WS 2002. LNCS, vol. 2593, pp. 169–183. Springer, Heidelberg (2003)
16. Morcillo, P.J., Moreno, G.: Programming with Fuzzy Logic Rules by using the FLOPER Tool. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2008. LNCS, vol. 5321, pp. 119–126. Springer, Heidelberg (2008)
17. Morcillo, P.J., Moreno, G., Penabad, J., Vázquez, C.: A Practical Management of Fuzzy Truth Degrees using FLOPER. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 20–34. Springer, Heidelberg (2010)
18. Ueng, P., Skrbic, S.: Implementing xquery fuzzy extensions using a native xml database. In: 2012 IEEE 13th International Symposium on Computational Intelligence and Informatics (CINTI), pp. 305–309. IEEE (2012)
19. Üstünkaya, E., Yazici, A., George, R.: Fuzzy data representation and querying in xml database. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 15(suppl. 01), 43–57 (2007)

Flexible Querying for SPARQL

Andrea Calì^{1,2}, Riccardo Frosini¹, Alexandra Poulovassilis¹,
and Peter T. Wood¹

¹ Dept. of Computer Science and Inf. Systems, Birkbeck University of London, UK

² Oxford-Man Institute of Quantitative Finance, University of Oxford, UK

{andrea,riccardo,ap,ptw}@dcs.bbk.ac.uk

Abstract. Flexible querying techniques can be used to enhance users' access to heterogeneous data sets, such as Linked Open Data. This paper extends SPARQL 1.1 with approximation and relaxation operators that can be applied to regular expressions for querying property paths in order to find more answers than would be returned by the exact form of a user query. We specify the semantics of the extended language and we consider the complexity of query answering with the new operators, showing that both data and query complexity are not impacted by our extensions. We present a query evaluation algorithm that returns results incrementally according to their “distance” from the original query. We have implemented this algorithm and have conducted preliminary trials over the YAGO SPARQL endpoint and the Lehigh University Benchmark, showing promising performance for the language extensions.

1 Introduction

Flexible querying techniques are used to enhance access to information stored within information systems, including in terms of user interaction. In particular, users querying an RDF dataset are not always aware of how a query should be formulated in order to correctly retrieve the desired answers. This problem can be caused by a lack of knowledge about the schema of the dataset or about the URIs used in the dataset; moreover, both schema and URIs can change over time. For example, suppose a user wishes to find events which took place in London on 12th December 2012 and poses the query $(x, \text{on}, "12/12/12") \text{ AND } (x, \text{in}, "London")$. This returns no results from the YAGO knowledge base because there are no property edges named “on” or “in”. Approximating “on” by “happenedOnDate” (which does appear in YAGO) and “in” by “happenedIn” still returns no answers, since “happenedIn” does not connect event instances directly to literals such as “London”. However, relaxing $(x, \text{happenedIn}, "London")$ to $(x, \text{type}, \text{Event})$ (using knowledge encoded in YAGO that the domain of “happenedIn” is *Event*) will return all events that occurred on 12th December 2012, including those occurring in London. Alternatively, instead of relaxing the second triple, another approximation step can be applied to $(x, \text{happenedIn}, "London")$, inserting the property edge *label* that connects URIs to their labels and yielding the query

$$(x, \text{happenedOnDate}, "12/12/12") \text{ AND } (x, \text{happenedIn}/\text{label}, "London")$$

This query now returns every event that occurred on 12th December 2012 in London.

SPARQL is the most prominent RDF query language and, since the latest extension of SPARQL 1.1, it supports property path queries¹ (i.e. *regular path* queries). In this paper we investigate how to extend SPARQL 1.1 with query approximation and query relaxation operations such as those illustrated in the above examples, calling the extended language SPARQL^{AR}. We study the computational complexity of the query answering problem; in particular, we show that the introduction of the new operators does not increase the computational complexity of the original language. We provide tight complexity bounds for several SPARQL fragments; we study *data complexity* (with only the instance graph as input), *query complexity* (with only the query as input) and *combined complexity* (with both query and instance as input). Our complexity results are summarised in Figure 3 on page 485. We then provide a query answering algorithm based on query rewriting, and present and discuss some preliminary experimental results.

Example. Suppose the user wishes to find the geographic coordinates of the “Battle of Waterloo” event by posing the query $(\langle \text{Battle_of_Waterloo} \rangle, \text{happenedIn}/(\text{hasLongitude}|\text{hasLatitude}), x)$. We see that this query uses the property paths extension of SPARQL, specifically the concatenation (/) and disjunction (|) operators. In the query, the property edge “happenedIn” is concatenated with either “hasLongitude” or “hasLatitude”, thereby finding a connection in the dataset between the event and its location (in our case Waterloo) and from the location to both its coordinates. This query does not return any answers from YAGO since YAGO does not store the geographic coordinates of Waterloo. However, by applying an approximation step, we can insert “isLocatedIn” after “happenedIn” which connects the URI representing Waterloo with the URI representing Belgium. The resulting query is

$\text{Battle_of_Waterloo}, \text{happenedIn}/\text{isLocatedIn}/(\text{hasLongitude}|\text{hasLatitude}), x$.

Since YAGO does have the geographic coordinates of Belgium, this query will return some answers that may be relevant for the user. Moreover, YAGO does store the coordinates of the “Battle of Waterloo” event, so if the query processor applies an approximation step that deletes the property edge “happenedIn”, instead of adding “isLocatedIn”, the resulting query $(\langle \text{Battle_of_Waterloo} \rangle, (\text{hasLongitude}|\text{hasLatitude}), x)$ returns the desired answers.

Related work. There have been different approaches to applying flexible querying to the Semantic Web. Most of these use similarity measures to retrieve additional relevant answers. An example of flexible querying with similarity measures can be found in [5], where the authors use matching functions for constants such as *strings* or *numeric values*. Similarly in [8] the authors have developed an extension of SPARQL called iSPARQL (imprecise SPARQL) which computes

¹ <http://www.w3.org/TR/sparql11-property-paths/>

string similarity matching using three different functions. A similarity measure technique which exploits the structure of the RDF dataset can be found in [4], where the authors navigate the RDF dataset as a graph in which every path is matched with respect to the query. Other techniques such as ontology driven similarity measures have been developed in [7,6,12]. These techniques use the RDFS ontology to retrieve extra answers and assign a score value to such answers. Finally, [11] shows how a conjunctive regular path query language can be effectively extended with approximation and relaxation techniques, using similar notions of approximation and relaxation as we use here.

In contrast to the above work, we focus here on the SPARQL 1.1 language. We extend, for the first time, this language with query approximation and relaxation operators, terming the extended language SPARQL^{AR} . We specify the semantics of SPARQL^{AR} , study the complexity of query answering, present a query evaluation algorithm returning answers ranked according to their “distance” from the original query, and present the results of a preliminary query performance study. Compared to [11], we focus here on SPARQL 1.1, derive new complexity results, provide a query rewriting algorithm, and present query performance results.

2 Preliminaries

In this section we give preliminary definitions needed to describe the syntax and semantics of SPARQL queries extended with regular expression patterns (known as ‘property paths’ in the SPARQL documentation) and flexible constructs, namely, query approximation and relaxation. For this kind of querying we will avoid blank nodes, since their use is discouraged for Linked Data because they represent a resource without specifying its name and are identified by an ID which may not be unique in the dataset [2]. Therefore we modify the definition of triples from [9].

Definition 1 (Sets, triples and variables). *Assume there are pairwise disjoint infinite sets U and L (URIs and literals). A tuple $\langle s, p, o \rangle \in U \times U \times (U \cup L)$ is called an RDF triple. In this tuple, s is the subject, p the predicate and o the object. We also assume an infinite set V of variables disjoint from the above sets. We abbreviate any union of these sets as, for instance, $UL = U \cup L$*

To accomodate our formalisation of flexible querying, we add weights to the edges of an RDF-Graph (changing again the definition from [9]). Initially these weights are all 0.

Definition 2 (RDF-Graph). *An RDF-Graph $G = (N, D, E)$ is defined as a finite set of nodes N such that $N \subset UL$, a finite set of predicates D used in the graph, where $D \subset U$, and a finite set of labelled weighted edges E , where each edge is of the form $\langle\langle s, p, o \rangle, c \rangle$ with subject $s \in N \cap U$, object $o \in N$, predicate $p \in D$ and c being the weight, or cost, of the edge.*

We discuss in the next section our query relaxation operator, which is based on the RDF-Schema (RDFS) data modelling vocabulary representing the ontology of an RDF dataset.

Definition 3 (RDF-Schema). *An ontology $K = (N_K, E_K)$ is a directed graph where each node in N_K represents either a class or a property, and each edge E_K is labelled with a symbol from the set $\{sc, sp, dom, range\}$. These edge labels in E_K encompass a fragment of the RDFS vocabulary, namely `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain`, `rdfs:range`.*

In an RDF-graph $G = (N, D, E)$, each node in N represents an instance or a class and each edge in E a property. In an ontology $K = (N_K, E_K)$, each node in N_K represents a class (a “class node”) or a property (a “property node”). The intersection of N and N_K is contained in the set of class nodes of K . D is contained in the set of property nodes of K . The predicate `type`, representing the RDF vocabulary `rdf:type` can be used in G to connect an instance of a class to a node representing that class.

Finally we define the notion of triple patterns, needed to construct queries, and mappings. Again we modify the definitions from [9] to exclude blank nodes.

Definition 4 (Triple patterns). *A triple pattern is a tuple $\langle x, z, y \rangle \in UV \times UV \times UVL$. Given a triple pattern $\langle x, z, y \rangle$, $\text{var}(\langle x, z, y \rangle)$ is the set of variables occurring in it.*

Definition 5 (Mapping). *A mapping μ from ULV to UL is a partial function $\mu : ULV \rightarrow UL$. We assume that $\mu(x) = x$ for all $x \in UL$ i.e. μ maps URIs and literals to themselves. The set $\text{var}(\mu)$, is the subset of V on which μ is defined. Given a triple pattern $\langle x, z, y \rangle$ and a mapping μ such that $\text{var}(\langle x, z, y \rangle) \subseteq \text{var}(\mu)$, $\mu(\langle x, z, y \rangle)$ is the triple obtained by replacing the variables in $\langle x, z, y \rangle$ by their image according to μ .*

2.1 Query Syntax

For regular expression patterns in SPARQL^{AR} we will use the definitions given in [3], conforming to the W3C syntax².

Definition 6 (Regular expression pattern). *A regular expression pattern $P \in \text{RegEx}(U)$ is defined as follows:*

$$P := \epsilon \mid - \mid p \mid (P_1 | P_2) \mid (P_1 / P_2) \mid P^*$$

where ϵ represents the empty pattern, $p \in U$ and $-$ is a symbol that denotes the disjunction of all URIs in U .

The syntax of query patterns is based on that of [3] but also includes, in our case, the query approximation and relaxation operators APPROX and RELAX:

² <http://www.w3.org/TR/sparql11-property-paths/>

Definition 7 (Query Pattern). A SPARQL^{AR} query pattern Q is defined as follows:

$$Q := UV \times V \times UVL \mid UV \times \text{RegEx}(U) \times UVL \mid Q_1 \text{ AND } Q_2 \mid Q \text{ FILTER } R \mid \text{RELAX}(UV \times \text{RegEx}(U) \times UVL) \mid \text{APPROX}(UV \times \text{RegEx}(U) \times UVL)$$

where R is a SPARQL built-in condition and Q, Q_1, Q_2 are query patterns. We denote by $\text{var}(Q)$ the set of all variables occurring in Q .

In the W3C SPARQL syntax, a dot (.) is used as the AND operator, but we avoid it for clarity and use AND instead. Note also that ϵ and $_$ cannot be specified in property paths in SPARQL 1.1.

A SPARQL query has the form $\text{SELECT}_{\vec{w}} \text{ WHERE } Q$, with $\vec{w} \subseteq \text{var}(Q)$ (we may omit here the keyword WHERE for simplicity). Given $Q' = \text{SELECT}_{\vec{w}} Q$, the head of the query, $\text{head}(Q')$, is \vec{w} if $\vec{w} \neq \emptyset$ and $\text{var}(Q)$ otherwise.

3 Semantics of SPARQL^{AR}

The semantics of SPARQL including regular expression query patterns is defined in [3]. For SPARQL^{AR} to handle the weight/cost of edges in an RDF-Graph and subsequently the cost of the approximation and relaxation operators (which we will describe in the following sections), we need to extend the notion of SPARQL query evaluation from returning a set of mappings to returning a set of pairs of the form $\langle \mu, \text{cost} \rangle$ where μ is a mapping and cost is its cost.

Two mappings μ_1 and μ_2 are said to be *compatible* if $\forall x \in \text{var}(\mu_1) \cap \text{var}(\mu_2)$, $\mu_1(x) = \mu_2(x)$. The *union* of two mappings $\mu = \mu_1 \cup \mu_2$ can be computed only if μ_1 and μ_2 are compatible. The resulting μ is a mapping where $\text{var}(\mu) = \text{var}(\mu_1) \cup \text{var}(\mu_2)$ and: for each x in $\text{var}(\mu_1) \cap \text{var}(\mu_2)$, we have $\mu(x) = \mu_1(x) = \mu_2(x)$; for each x in $\text{var}(\mu_1)$ but not in $\text{var}(\mu_2)$, we have $\mu(x) = \mu_1(x)$; and for each x in $\text{var}(\mu_2)$ but not in $\text{var}(\mu_1)$, we have $\mu(x) = \mu_2(x)$.

We next define the *union* and *join* of two sets of query evaluation results, M_1 and M_2 :

$$M_1 \cup M_2 = \{ \langle \mu, \text{cost} \rangle \mid \langle \mu, \text{cost}_1 \rangle \in M_1 \text{ or } \langle \mu, \text{cost}_2 \rangle \in M_2 \text{ with } \text{cost} = \text{cost}_1 \text{ if } \nexists \text{cost}_2. \langle \mu, \text{cost}_2 \rangle \in M_2, \text{cost} = \text{cost}_2 \text{ if } \nexists \text{cost}_1. \langle \mu, \text{cost}_1 \rangle \in M_1, \text{ and } \text{cost} = \min(\text{cost}_1, \text{cost}_2) \text{ otherwise} \}.$$

$$M_1 \bowtie M_2 = \{ \langle \mu_1 \cup \mu_2, \text{cost}_1 + \text{cost}_2 \rangle \mid \langle \mu_1, \text{cost}_1 \rangle \in M_1 \text{ and } \langle \mu_2, \text{cost}_2 \rangle \in M_2 \text{ with } \mu_1 \text{ and } \mu_2 \text{ compatible mappings} \}.$$

3.1 Exact Semantics

The semantics of a triple pattern t that may include regular expression patterns as its second component, with respect to a graph G , denoted $[[t]]_G$, is defined recursively as follows:

$$\begin{aligned}
[[\langle x, \epsilon, y \rangle]]_G &= \{\langle \mu, 0 \rangle \mid \text{var}(\mu) = \text{var}(\langle x, \epsilon, y \rangle) \wedge \exists c \in N . \mu(x) = \mu(y) = c\} \\
[[\langle x, z, y \rangle]]_G &= \{\langle \mu, \text{cost} \rangle \mid \text{var}(\mu) = \text{var}(\langle x, z, y \rangle) \wedge \langle \mu(\langle x, z, y \rangle), \text{cost} \rangle \in E\} \\
[[\langle x, P_1 | P_2, y \rangle]]_G &= [[\langle x, P_1, y \rangle]]_G \cup [[\langle x, P_2, y \rangle]]_G \\
[[\langle x, P_1 / P_2, y \rangle]]_G &= [[\langle x, P_1, z \rangle]]_G \bowtie [[\langle z, P_2, y \rangle]]_G \\
[[\langle x, P^*, y \rangle]]_G &= [[\langle x, \epsilon, y \rangle]]_G \cup [[\langle x, P, y \rangle]]_G \cup \bigcup_{n \geq 1} \{\langle \mu, \text{cost} \rangle \mid \langle \mu, \text{cost} \rangle \in \\
&\quad [[\langle x, P, z_1 \rangle]]_G \bowtie [[\langle z_1, P, z_2 \rangle]]_G \bowtie \cdots \bowtie [[\langle z_n, P, y \rangle]]_G\}
\end{aligned}$$

where P , P_1 , P_2 are regular expression patterns, x , y , z are in ULV , and z, z_1, \dots, z_n are fresh variables.

A mapping *satisfies a condition R*, denoted $\mu \models R$, as follows:

- R is $x = c$: $\mu \models R$ if $x \in \text{var}(\mu)$, $c \in L$ and $\mu(x) = c$
- R is $x = y$: $\mu \models R$ if $x, y \in \text{var}(\mu)$ and $\mu(x) = \mu(y)$
- R is $\text{isURI}(x)$: $\mu \models R$ if $x \in \text{var}(\mu)$ and $\mu(x) \in U$
- R is $\text{isLiteral}(x)$: $\mu \models R$ if $x \in \text{var}(\mu)$ and $\mu(x) \in L$
- R is $R_1 \wedge R_2$: $\mu \models R$ if $\mu \models R_1$ and $\mu \models R_2$
- R is $R_1 \vee R_2$: $\mu \models R$ if $\mu \models R_1$ or $\mu \models R_2$
- R is $\neg R_1$: $\mu \models R$ if it is not the case that $\mu \models R_1$

Given the above definitions, the overall semantics of queries (excluding AP-PROX and RELAX) is as follows, where Q , Q_1 , Q_2 are query patterns and the projection operator $\pi_{\vec{w}}$ selects only the subsets of the mappings relating to the variables in \vec{w} :

$$\begin{aligned}
[[Q_1 \text{ AND } Q_2]]_G &= [[Q_1]]_G \bowtie [[Q_2]]_G \\
[[Q \text{ FILTER } R]]_G &= \{\langle \mu, \text{cost} \rangle \in [[Q]]_G \mid \mu \models R\} \\
[[\text{SELECT}_{\vec{w}} Q]]_G &= \pi_{\vec{w}}([[Q]]_G)
\end{aligned}$$

We will omit the SELECT keyword from a query Q if $\vec{w} = \text{vars}(Q)$.

3.2 Query Relaxation

Our relaxation operator is based on that in [11] and relies on RDFS *entailment*. We give a summary here and refer the reader to that paper for full details.

An RDFS graph K_1 entails an RDFS graph K_2 , denoted $K_1 \models_{RDFS} K_2$, if we can derive K_2 by applying the rules in Figure 1 iteratively to K_1 . For the fragment of RDFS that we consider, $K_1 \models_{RDFS} K_2$ if and only if $K_2 \subseteq cl(K_1)$, with $cl(K_1)$ being the closure of the RDFS Graph K_1 under these rules.

In order to apply relaxation to queries, we need to define the *extended reduction* of an ontology K . Given an ontology K , its extended reduction $extRed(K)$ is computed as follows: (i) compute $cl(K)$; (ii) apply rules of Figure 2 in reverse until no longer applicable (applying a rule in reverse means deleting a triple deducible by the rule); (iii) apply rules 1 and 3 of Figure 1 in reverse until no longer

applicable. Henceforth, we assume that $K = \text{extRed}(K)$, which allows us to perform *direct* relaxations on queries (see below) that correspond to the ‘smallest’ relaxation steps. This is necessary if we are to return query answers to users incrementally in order of increasing cost. We also need K to be acyclic in order for direct relaxation to be well-defined. We say that a triple pattern $\langle x, p, y \rangle$

$$\begin{array}{ll} \text{Subproperty (1)} & \frac{(a, sp, b)(b, sp, c)}{(a, sp, c)} \quad (2) \frac{(a, sp, b)(x, a, y)}{(x, b, y)} \\ \text{Subclass (3)} & \frac{(a, sc, b)(b, sc, c)}{(a, sc, c)} \quad (4) \frac{(a, sc, b)(x, type, a)}{(x, type, b)} \\ \text{Typing (5)} & \frac{(a, dom, c)(x, a, y)}{(x, type, c)} \quad (6) \frac{(a, range, d)(x, a, y)}{(y, type, d)} \end{array}$$

Fig. 1. RDFS entailment rules

$$\begin{array}{ll} (\text{e1}) & \frac{(b, dom, c)(a, sp, b)}{(a, dom, c)} \quad (\text{e2}) \frac{(b, range, c)(a, sp, b)}{(a, range, c)} \\ (\text{e3}) & \frac{(a, dom, b)(b, sc, c)}{(a, dom, c)} \quad (\text{e4}) \frac{(a, range, b)(b, sc, c)}{(a, range, c)} \end{array}$$

Fig. 2. Additional rules for extended reduction of an RDFS ontology

directly relaxes to a triple pattern $\langle x', p', y' \rangle$, denoted $\langle x, p, y \rangle \prec_i \langle x', p', y' \rangle$, if $\text{vars}(\langle x, p, y \rangle) = \text{vars}(\langle x', p', y' \rangle)$ and applying rule i from Figure 1 we can derive $\langle x', p', y' \rangle$ from $\langle x, p, y \rangle$. There is a cost c_i associated with the application of a rule i . We note that since rule 6 changes the position of y , which we want to avoid when it comes to relaxing regular expression patterns (see below), we actually use (d, type^-, y) as the consequent of rule 6; and we also allow a modified form of rule 4 where the triples involving *type* appear with their arguments in reverse order and *type* is replaced by *type*⁻.

We say that a triple pattern $\langle x, p, y \rangle$ *relaxes to* a triple pattern $\langle x', p', y' \rangle$, denoted $\langle x, p, y \rangle \leq_K \langle x', p', y' \rangle$, if starting from $\langle x, p, y \rangle$ there is a sequence of direct relaxations that derives $\langle x', p', y' \rangle$. The relaxation cost of deriving $\langle x, p, y \rangle$ from $\langle x', p', y' \rangle$, denoted $\text{rcost}(\langle x, p, y \rangle, \langle x', p', y' \rangle)$, is the minimum cost of applying such a sequence of direct relaxations.

We now define the semantics of the RELAX operator in SPARQL^{AR} as follows:

$$\begin{aligned} [[\text{RELAX}(x, p, y)]]_G &= [[\langle x, p, y \rangle]]_G \cup \{ \langle \mu, \text{cost} + \text{rcost}(\langle x, p, y \rangle, \langle x', p', y' \rangle) \rangle \mid \\ &\quad \langle x, p, y \rangle \leq_K \langle x', p', y' \rangle \wedge \langle \mu, \text{cost} \rangle \in [[\langle x', p', y' \rangle]]_G \} \\ [[\text{RELAX}(x, P_1 | P_2, y)]]_G &= [[\text{RELAX}(x, P_1, y)]]_G \cup [[\text{RELAX}(x, P_2, y)]]_G \\ [[\text{RELAX}(x, P_1 / P_2, y)]]_G &= [[\text{RELAX}(x, P_1, z)]]_G \bowtie [[\text{RELAX}(z, P_2, y)]]_G \\ [[\text{RELAX}(x, P^*, y)]]_G &= [[\langle x, \epsilon, y \rangle]]_G \cup [[\text{RELAX}(x, P, y)]]_G \cup \\ &\quad \bigcup_{n \geq 1} \{ \langle \mu, \text{cost} \rangle \mid \langle \mu, \text{cost} \rangle \in [[\text{RELAX}(x, P, z_1)]]_G \bowtie \\ &\quad \bowtie [[\text{RELAX}(z_1, P, z_2)]]_G \bowtie \cdots \bowtie [[\text{RELAX}(z_n, P, y)]]_G \} \end{aligned}$$

where P , P_1 , P_2 are regular expression patterns, x , x' , y , y' are in ULV , p , p' are in U , and z , z_1, \dots, z_n are fresh variables.

3.3 Query Approximation

Regarding query approximation, we consider a set of edit operations which transform a regular expression pattern P into a new expression pattern P' . We focus here on the edit operations *deletion*, *insertion* and *substitution* (leaving other possible edit operations such as *transposition* and *inversion* for future work) which are defined as follows:

$$\begin{array}{ll} A/p/B \rightsquigarrow (A/\epsilon/B) & \text{deletion} \\ A/p/B \rightsquigarrow (A/-/B) & \text{substitution} \\ A/p/B \rightsquigarrow (A/-/p/B) & \text{left insertion} \\ A/p/B \rightsquigarrow (A/p/-/B) & \text{right insertion} \end{array}$$

Here, A and B denote any regular expression and the symbol $-$ represents every URI from U — so the edit operations allow us to insert any URI and substitute a URI by any other in U . The application of an edit operation op has a cost c_{op} associated with it.

We can apply the above set of rules to a URI p in order to approximate it to a regular expression P . We write $p \rightsquigarrow^* P$ if we can apply a sequence of edit operations to p to derive P . The edit cost of deriving P from p , denoted $ecost(p, P)$, is the minimum cost of applying such a sequence of edit operations.

We now define the semantics of the APPROX operator in SPARQL^{AR} as follows:

$$\begin{aligned} [[\text{APPROX}(x, p, y)]]_G &= [[\langle x, p, y \rangle]]_G \cup \bigcup \{ \langle \mu, cost + ecost(p, P) \rangle \mid \\ &\quad p \rightsquigarrow^* P \wedge \langle \mu, cost \rangle \in [[\langle x, P, y \rangle]]_G \} \\ [[\text{APPROX}(x, P_1 | P_2, y)]]_G &= [[\text{APPROX}(x, P_1, y)]]_G \cup [[\text{APPROX}(x, P_2, y)]]_G \\ [[\text{APPROX}(x, P_1 / P_2, y)]]_G &= [[\text{APPROX}(x, P_1, z)]]_G \bowtie [[\text{APPROX}(z, P_2, y)]]_G \\ [[\text{APPROX}(x, P^*, y)]]_G &= [[\langle x, \epsilon, y \rangle]]_G \cup [[\text{APPROX}(x, P, y)]]_G \cup \\ &\quad \bigcup_{n \geq 1} \{ \langle \mu, cost \rangle \mid \langle \mu, cost \rangle \in [[\text{APPROX}(x, P, z_1)]]_G \bowtie \\ &\quad \bowtie [[\text{APPROX}(z_1, P, z_2)]]_G \bowtie \cdots \bowtie [[\text{APPROX}(z_n, P, y)]]_G \} \end{aligned}$$

where P , P_1 , P_2 are regular expression patterns, x , y are in ULV , p , p' are in U , and z , z_1, \dots, z_n are fresh variables.

4 Complexity of Query Answering

In this section we study the combined, data and query complexity of SPARQL including regular expression patterns, the new APPROX and RELAX operators

and weighted edges in the RDF graph. Our work extends the complexity results in [9,10,13] for simple SPARQL queries and in [1] for SPARQL with regular expression patterns to include our new flexible query constructs in SPARQL^{LR}.

The complexity of query evaluation is based on the following decision problem, which we denote EVALUATION: given as input a graph G , a query Q and a pair $\langle \mu, cost \rangle$, is it the case that $\langle \mu, cost \rangle \in [[Q]]_G$?

Queries without regular expression patterns (i.e. where there are only triple patterns of the form $UV \times UV \times UVL$) and without the flexible query constructs, can be evaluated in polynomial time if they include only the operators AND and FILTER. This result can be achieved by adapting the algorithm given in [9,10,13], adding the cost of the mappings in our setting:

Theorem 1. *EVALUATION can be solved in time $O(|E| \cdot |Q|)$ for query pattern expressions constructed using only AND and FILTER operators.*

Proof. We give an algorithm for the EVALUATION problem that runs in polynomial time: first, for each i such that the triple pattern $\langle x, z, y \rangle_i$ is in Q , we verify that $\langle \mu(\langle x, z, y \rangle_i), cost_i \rangle \in E$ for some $cost_i$. If this is not the case, or if $\sum_i cost_i \neq cost$ we return False. Otherwise we check if μ satisfies the FILTER condition and return True or False accordingly. It is evident that the algorithm runs in polynomial time since verifying that $\langle \mu(\langle x, z, y \rangle_i), cost_i \rangle \in E$ can be done in time $|E|$.

When we add regular expression patterns to queries, the complexity of query evaluation increases slightly. To show this, we start by building an NFA $M_P = (S, T)$ that recognises $\mathcal{L}(P)$, the language denoted by the regular expression P , where S is the set of states (including s_0 and s_f representing the initial and final states respectively) and T is the set of transitions, each of cost 0. We then construct the weighted *product automaton*, H , of G and M_P as follows:

- The states of H are the Cartesian product of the set of nodes N of G and the set of states S of M_P .
- For each transition $\langle\langle s, p, s' \rangle, 0 \rangle$ in M_P and each edge $\langle\langle a, p, b \rangle, cost \rangle$ in E , there is a transition $\langle\langle s, a \rangle, \langle s', b \rangle, cost \rangle$ in H .

Then we check if there exists a path from $\langle s_0, \mu(x) \rangle$ to $\langle s_f, \mu(y) \rangle$ in H . In case there is more than one path, we select one with the minimum cost using Dijkstra's algorithm. Knowing that the number of nodes in H is equal to $|N| \cdot |S|$, the number of edges is at most $|E| \cdot |T|$, and that $|T| \leq |S|^2$, the evaluation can be performed in time $O(|E| \cdot |S|^2 + |N| \cdot |S| \cdot \log(|N| \cdot |S|))$. Noting that the query size is proportional to $|S|$, on the assumption that $|E| \geq |N|$ we therefore have:

Theorem 2. *EVALUATION can be solved in time $O(|E| \cdot |Q|^2)$ for query pattern expressions constructed using only AND, FILTER and regular expression patterns.*

Next, we discuss how adding the SELECT operator increases the complexity even if FILTER is excluded (i.e. including costs in the graph G and regular expression patterns, the complexity given by [9,10] does not change):

Theorem 3. *EVALUATION* is NP-complete for query pattern expressions constructed using only AND and regular expression patterns, and including the projection operator SELECT.

Proof. We first show that the evaluation problem is in NP. Given a pair $\langle \mu, cost \rangle$ and a query $\text{SELECT}_{\vec{w}} Q$ where Q does not include FILTER, we have to check whether $\langle \mu, cost \rangle$ is in $[[\text{SELECT}_{\vec{w}} Q]]_G$. We can guess a new mapping μ' such that $\pi_{\vec{w}}(\langle \mu', cost \rangle) = \langle \mu, cost \rangle$ and consequently check that $\langle \mu', cost \rangle \in [[Q]]_G$ (which can be done in polynomial time as we have seen in Theorem 2). The number of guesses is bounded by the number of variables in Q and values from G to which they can be mapped.

For NP-hardness we first define the problem of graph 3-colourability, which is known to be NP-complete: given a graph $\Gamma = (N_\Gamma, E_\Gamma)$ and three colours r, g, b , is it possible to assign a colour to each node in N_Γ such that no pair of nodes connected by an edge in E_Γ are of the same colour?

We next define the following RDF graph $G = (N, D, E)$:

$$\begin{aligned} N &= \{r, g, b, a\} & D &= \{a, p\} \\ E &= \{\langle r, p, g \rangle, 0\rangle, \langle r, p, b \rangle, 0\rangle, \langle g, p, b \rangle, 0\rangle, \langle g, p, r \rangle, 0\rangle, \\ &\quad \langle b, p, r \rangle, 0\rangle, \langle b, p, g \rangle, 0\rangle, \langle a, a, a \rangle, 0\rangle\} \end{aligned}$$

Now we construct the following query Q such that there is a variable x_i corresponding to each node n_i of Γ and there is a triple pattern of the form $\langle x_i, p, x_j \rangle$ in Q if and only if there is an edge (n_i, n_j) in Γ :

$$Q = \text{SELECT}_x((x_i, p, x_j) \text{ AND } \dots \text{ AND } (x'_i, p, x'_j) \text{ AND } (a, a, x))$$

It is easy to verify that the graph Γ is colourable if and only if $\langle \mu, 0 \rangle \in [[Q]]_G$ with $\mu = \{x \rightarrow a\}$.

The following lemma will help us to show that adding the APPROX and RELAX operators will not increase the complexity.

Lemma 1. *EVALUATION of $[[\text{APPROX}(x, P, y)]]_G$ and $[[\text{RELAX}(x, P, y)]]_G$ can be done in polynomial time.*

Proof (Premise). Given a pair $\langle \mu, cost \rangle$ we have to verify in polynomial time that $\langle \mu, cost \rangle \in [[\text{APPROX}(x, P, y)]]_G$ or $\langle \mu, cost \rangle \in [[\text{RELAX}(x, P, y)]]_G$. We start by building an NFA $M_P = (S, T)$ as described earlier.

Proof (Approximation). An approximate automaton $A_P = (S, T')$ is constructed starting from M_P and adding the following additional transitions (similarly to the construction in [11]):

- For each state $s \in S$ there is a transition $\langle \langle s, _, s \rangle, \alpha \rangle$, where α is the cost of insertion.
- For each transition $\langle \langle s, p, s' \rangle, 0 \rangle$ in M_P , where $p \in D$, there is a transition $\langle \langle s, \epsilon, s' \rangle, \beta \rangle$, where β is the cost of deletion.

- For each transition $\langle\langle s, p, s' \rangle, 0 \rangle$ in M_P , where $p \in D$, there is a transition $\langle\langle s, _, s' \rangle, \gamma \rangle$, where γ is the cost of substitution.

We then form the weighted product automaton, H , of G and A_P as follows:

- The states of H will be the Cartesian product of the set of nodes N of G and the set of states S of A_P .
- For each transition $\langle\langle s, p, s' \rangle, cost_1 \rangle$ in A_P and each edge $\langle\langle a, p, b \rangle, cost_2 \rangle$ in E , there is a transition $\langle\langle s, a, \langle s', b \rangle, cost_1 + cost_2 \rangle$ in H .
- For each transition $\langle\langle s, \epsilon, s' \rangle, cost \rangle$ in A_P and each node $a \in N$, there is a transition $\langle\langle s, a, \langle s', a \rangle, cost \rangle$ in H .
- For each transition $\langle\langle s, _, s' \rangle, cost_1 \rangle$ in A_P and each edge $\langle\langle a, p, b \rangle, cost_2 \rangle$ in E , there is a transition $\langle\langle s, a, \langle s', b \rangle, cost_1 + cost_2 \rangle$ in H .

Finally we check if there exists a path from $\langle s_0, \mu(x) \rangle$ to $\langle s_f, \mu(y) \rangle$ in H . Again, if there exists more than one path we select one with minimum cost using Dijkstra's Algorithm. Knowing that the number of nodes in H is $|N| \cdot |S|$ and that the number of edges in H is at most $(|E| + |N|) \cdot |S|^2$, the evaluation can therefore be computed in $O((|E| + |N|) \cdot |S|^2 + |N| \cdot |S| \cdot \log(|N| \cdot |S|))$.

Proof (Relaxation). Given an ontology $K = extRed(K)$ we build the *relaxed automaton* $R_P = (S', T', S_0, S_f)$ starting from M_P (similarly to the construction in [11]). S_0 and S_f represent the sets of initial and final states, and S' contains every state in S plus the states in S_0 and S_f . Initially S_0 and S_f contain s_0 and s_f respectively. Each initial and final state in S_0 and S_f is labelled with either a constant or the symbol $*$; in particular, s_0 is labelled with x if x is a constant or $*$ if it is a variable and similarly s_f is labelled with y if y is a constant or $*$ if it is a variable. An *incoming (outgoing) clone* of a state s is a new state s' such that s' is an initial or final state if s is, s' has the same set of incoming (outgoing) transitions as s , and has no outgoing (incoming) transitions. Initially T' contains all the transitions in T . We recursively add states to S_0 and S_f , and transitions to T' as follows until we reach a fixpoint:

- For each transition $\langle\langle s, p, s' \rangle, cost \rangle \in T'$ and $\langle p, sp, p' \rangle \in K$ add the transition $\langle\langle s, p', s' \rangle, cost + \alpha \rangle$ to T' , where α is the cost of applying rule 2.
- For each transition $\langle\langle s, type, s' \rangle, cost \rangle \in T'$, $s' \in S_f$ and $\langle c, sc, c' \rangle \in K$ such that s' is annotated with c add an outgoing clone s'' of s' annotated with c' to S_f and add the transition $\langle\langle s, type, s'' \rangle, cost + \beta \rangle$ to T' , where β is the cost of applying rule 4.
- For each transition $\langle\langle s, type^-, s' \rangle, cost \rangle \in T'$, $s \in S_0$ and $\langle c, sc, c' \rangle \in K$ such that s is annotated with c add an incoming clone s'' of s annotated with c' to S_0 and add the transition $\langle\langle s'', type^-, s' \rangle, cost + \beta \rangle$ to T' , where β is the cost of applying rule 4.
- For each $\langle\langle s, p, s' \rangle, cost \rangle \in T'$, $s' \in S_f$ and $\langle p, dom, c \rangle$ such that s' is annotated with a constant, add an outgoing clone s'' of s' annotated with c to S_f , and add the transition $\langle\langle s, type, s'' \rangle, cost + \gamma \rangle$ to T' , where γ is the cost of applying rule 5.

- For each $\langle\langle s, p, s' \rangle, cost \rangle \in T'$, $s \in S_0$ and $\langle p, range, c \rangle$ such that s is annotated with a constant, add an incoming clone s'' of s annotated with c to S_0 , and add the transition $\langle\langle s'', type^-, s' \rangle, cost + \delta \rangle$ to T' , where δ is the cost of applying rule 6.

(We note that because queries and graphs do not contain edges labelled sc or sp , rules 1 and 3 in Figure 1 are inapplicable as far as query relaxation is concerned.)

We then form the weighted product automaton, H , of G and R_P as follows:

- For each node $a \in N$ of G and each state $s \in S'$ of R_P , then $\langle s, a \rangle$ is a state of H if s is labelled with either $*$ or a , or is unlabelled.
- For each transition $\langle\langle s, p, s' \rangle, cost_1 \rangle$ in R_P and each edge $\langle\langle a, p, b \rangle, cost_2 \rangle$ in E such that $\langle s, a \rangle$ and $\langle s', b \rangle$ are states of H , then there is a transition $\langle\langle s, a \rangle, \langle s', b \rangle, cost_1 + cost_2 \rangle$ in H .
- For each transition $\langle\langle s, type^-, s' \rangle, cost_1 \rangle$ in R_P and each edge $\langle\langle a, type, b \rangle, cost_2 \rangle$ in E such that $\langle s, b \rangle$ and $\langle s', a \rangle$ are states of H , then there is a transition $\langle\langle s, b \rangle, \langle s', a \rangle, cost_1 + cost_2 \rangle$ in H .

Finally we check if there exists a path from $\langle s, \mu(x) \rangle$ to $\langle s', \mu(y) \rangle$ in H , where $s \in S_0$ and $s' \in S_f$. Again, if there exists more than one path we select one with minimum cost using Dijkstra's Algorithm. Knowing that the number of nodes in H is at most $|N| \cdot |S'|$ and the number of edges in H is at most $|E| \cdot |S'|^2$, the evaluation can therefore be computed in $O(|E| \cdot |S'|^2 + |N| \cdot |S'| \cdot \log(|N| \cdot |S'|))$.

Proof (Conclusion). We can conclude that both query approximation and query relaxation can be evaluated in polynomial time. In particular, the evaluation can be done in $O(|E|)$ time with respect to the data and in polynomial time with respect to the query.

Through the previous lemma we can conclude our combined complexity study with the following theorem:

Theorem 4. *EVALUATION is NP-complete for query pattern expressions constructed using regular expression patterns and the operators AND, FILTER, RELAX, APPROX and SELECT.*

With the next theorem, we consider the behaviour of our extended SPARQL language in terms of data complexity. In particular we extend what we have already shown in Theorems 3 and 4 by changing the decision problem stated earlier to the following: given as input a graph G and a pair $\langle \mu, cost \rangle$, is it the case that $\langle \mu, cost \rangle \in [[Q]]_G$, with Q a fixed query?

Theorem 5. *EVALUATION is PTIME in data complexity for query pattern expressions constructed using regular expression patterns, and the operators AND, FILTER, RELAX, APPROX and SELECT.*

Proof. In order to prove the theorem, we devise an algorithm that runs in polynomial time with respect to the size of the graph G . We start by building a new mapping μ' such that each variable $x \in var(\mu')$ appears in $var(Q)$ but not in $var(\mu)$, and to each we assign a different constant from ND . We then verify in polynomial time that $\langle \mu \cup \mu', cost \rangle$ is in $[[Q]]_G$. The number of mappings we can generate is $O(|ND|^{|var(Q)|})$. Since the query is fixed we can therefore say that the evaluation with respect to the data is in polynomial time.

Results for the query complexity of our extended SPARQL language follow directly from Lemma 1 and Theorems 1, 2 and 3. In fact, Lemma 1 and Theorems 1 and 2 show an algorithm that evaluates AND queries with APPROX, RELAX and regular expression patterns in polynomial time with respect to the query. The proof of Theorem 3 reduces the problem of EVALUATION from the graph 3-colourability problem, and since the graph constructed for the reduction does not change with respect to the input problem, we can affirm that query complexity remains NP-complete.

We summarise the complexity study of our extended SPARQL language in Figure 3, where we show the combined, data and query complexity for the language fragments identified by the operators included.

Operators	Data Complexity	Query Complexity	Combined Complexity
AND, FILTER	$O(E)$	$O(Q)$	$O(E \cdot Q)$
AND, FILTER, RegEx	$O(E)$	$O(Q ^2)$	$O(E \cdot Q ^2)$
RELAX, APPROX	$O(E)$	P-Time	P-Time
RELAX, APPROX, AND, FILTER, RegEx	$O(E)$	P-Time	P-Time
AND, SELECT	P-Time	NP-Complete	NP-Complete
RELAX, APPROX, AND, FILTER, RegEx, SELECT	P-Time	NP-Complete	NP-Complete

Fig. 3. Complexity of various SPARQL fragments

5 Query Evaluation

In this section we describe how to compute the relaxed and approximated answer of a SPARQL^{AR} query by making use of a query rewriting algorithm, following a similar approach to [6, 7, 12]. In particular, given a query Q with the APPROX and/or RELAX operators, our goal is to incrementally build a set of queries $\{Q_0, \dots, Q_n\}$ that do not contain these operators such that $\bigcup_i [[Q_i]]_G = [[Q]]_G$. Moreover we need to produce answers in order of increasing cost.

The query rewriting algorithm starts by considering the query Q_0 which returns the exact answer of the query Q , i.e., ignoring the APPROX and RELAX operators. To keep track of which triple patterns need to be relaxed or approximated, we label such triple patterns with A for approximation and R for relaxation. For each triple pattern $\langle x_i, P_i, y_i \rangle$ in Q_0 labelled with A (R) and each URI p that appears in P_i , we construct a new query applying one step of approximation (relaxation) to p , and to each query we assign the cost of applying such a step. From each query constructed in this way, we next produce a new set of queries applying a second step of approximation or relaxation. The cost of each query is equal to the cost of the original query plus the cost of the sequence of approximations or relaxations applied to it. For practical reasons we limit the number of queries generated by bounding the cost of the queries up to a maximum value c .

To compute the query answers, we incrementally apply an evaluation function, $eval$, to each query generated by the rewriting algorithm in ranked order of the cost of the queries, and to each mapping returned by $eval$ we assign the cost of the query. If we generate a particular mapping more than once we keep the one with the lowest cost. The $eval$ function takes as input a query Q and a graph G and returns $[[Q]]_G$. The overall query evaluation algorithm is defined below where rew is the query rewriting algorithm and we assume that the set of mappings M is maintained in order of increasing cost (e.g. as a priority queue):

Algorithm 1. Flexible Query Evaluation

```

input : Query  $Q$ ; approx/relax max cost  $c$ ; Graph  $G$ ; Ontology  $K$ .
output: List of pairs mapping/cost  $M$  sorted by cost.
 $M := \emptyset$ ;
foreach  $\langle Q', cost \rangle \in rew(Q, c, K)$  do
  foreach  $\langle \mu, 0 \rangle \in eval(Q', G)$  do
     $M := M \cup \{\langle \mu, cost \rangle\}$ 
return  $M$ ;

```

Example 1. Consider the following ontology K (satisfying $K = extRed(K)$), which is a fragment of the YAGO knowledge base³ derived from multiple sources such as Wikipedia, WordNet and GeoNames:

$$\begin{aligned}
 K = & (\{happenedIn, placedIn, happenedOnDate, Event\}, \\
 & \{\langle happenedIn, sp, placedIn \rangle, \langle happenedIn, dom, Event \rangle\})
 \end{aligned}$$

Suppose a user wishes to find every event which took place in London on 12th December 2012 and poses the following query Q :

APPROX($x, happenedOnDate, "12/12/12"$) AND RELAX($x, happenedIn, "London"$).

Without applying APPROX or RELAX, this query does not return any answers when evaluated on the YAGO endpoint (because “happenedIn” connects to URIs representing places and “London” is a literal, not a URI). After the first step of approximation and relaxation, the following queries are generated:

³ <http://www.mpi-inf.mpg.de/yago-naga/yago/>

$$\begin{aligned}
 Q_1 &= (x, \epsilon, "12/12/12")_A \text{ AND } (x, \text{happenedIn}, "London")_R \\
 Q_2 &= (x, \text{happenedOnDate}/_, "12/12/12")_A \text{ AND } (x, \text{happenedIn}, "London")_R \\
 Q_3 &= (x, _\text{happenedOnDate}, "12/12/12")_A \text{ AND } (x, \text{happenedIn}, "London")_R \\
 Q_4 &= (x, _, "12/12/12")_A \text{ AND } (x, \text{happenedIn}, "London")_R \\
 Q_5 &= (x, \text{happenedOnDate}, "12/12/12")_A \text{ AND } (x, \text{placedIn}, "London")_R \\
 Q_6 &= (x, \text{happenedOnDate}, "12/12/12")_A \text{ AND } (x, \text{type}, \text{Event})_R
 \end{aligned}$$

Each of these also returns empty results, with the exception of query Q_6 which returns every event occurring on 12/12/12 (including amongst them the events occurring in London that are of interest to the user).

We have conducted a theoretical study of the correctness and termination of the Rewriting Algorithm which can be found in the extended version of the paper (<http://www.dcs.bbk.ac.uk/~riccardo/ODBASE2014Extended.pdf>), where the Rewriting Algorithm itself is also specified in detail.

6 Experimental Results

We have implemented our query evaluation algorithm and have conducted preliminary trials over the YAGO SPARQL endpoint and the Lehigh University Benchmark (LUBM)⁴. Using the latter we generated 3 datasets: D_1 with 149,973 triples (13Mb in XML format), D_2 with 421,562 triples (44Mb), and D_3 with 673,416 triples (65Mb). The LUBM ontology⁵ consists of 355 statements and describes a university domain. We ran our experiments on a Windows 7 computer with 4Gb of RAM and a quadcore-core i7 CPU at 2.0Ghz. The query evaluation algorithm was implemented in Java and we used Jena for the SPARQL query execution⁶.

In Figure 4 we show the number of answers and the number of seconds it takes to answer a set of exact queries, $Q_1 - Q_4$. In Figure 5 we show both the number of answers and the number of seconds it takes to answer approximated and/or relaxed versions of the same queries, $Q'_1 - Q'_4$. For these experiments, the cost of all edit and relaxation operations was set to 1 (in practice the user may set different costs depending on the query or application). The maximum cost was also fixed at 1 for each of $Q'_1 - Q'_4$ (in practice the user would set a small cost – 0 or 1 – to begin with, explore the results returned, and iteratively request more results at greater cost as necessary).

Query Q_1 , which returns every university and its head, is as follows:

```
SELECT ?X ?Z WHERE {?X ub:headOf ?Z}
```

For Q'_1 , we RELAX the triple pattern to find other people who work at a university. With the maximum cost set to 1, the rewriting algorithm generated only 1

⁴ <http://swat.cse.lehigh.edu/projects/lubm/>

⁵ <http://swat.cse.lehigh.edu/onto/univ-bench.owl>

⁶ <https://jena.apache.org/>

more query. The new query generated 817 answers at cost 1 for the first dataset, 2263 answers at cost 1 for the second and finally 3606 answers for the third.

Query Q_2 returns people who work for department `Department0.University0` with their email address and phone number:

```
SELECT *
WHERE {?X ub:worksFor <http://www.Department0.University0.edu> .
      ?X ub:name ?Y1 . ?X ub:emailAddress ?Y2 . ?X ub:telephone ?Y3}
```

For Q'_2 , we APPROX the first triple pattern, allowing details to be returned of people who have other relationships with the department. Even with a maximum cost of 1, Q'_2 returns many more answers than Q_2 .

Query Q_3 returns every sub-organization of organizations affiliated with `University0`:

```
SELECT ?Y ?Z
WHERE {?Y ub:subOrganizationOf* ?Z .
      ?Z ub:affiliatedOrganizationOf <http://www.University0.edu>}
```

For Q'_3 we RELAX the first triple pattern and APPROX the second one, allowing answers to be returned relating to organizations that have other relationships with `University0`.

Query Q_4 returns students along with courses they attend which are taught by their advisor:

```
SELECT ?X ?Z
WHERE {?X ub:advisor/ub:teacherOf ?Z . ?X ub:takesCourse ?Z}
```

For Q'_4 we RELAX the first triple pattern and APPROX the second one.

Dataset	Q_1	Q_2	Q_3	Q_4
D_1	23/0.001s	34/0.004s	0/0.197s	331/0.129s
D_2	63/0.006s	34/0.005s	0/0.64s	883/0.296s
D_3	100/0.007s	34/0.006s	0/1.67s	1381/0.517s

Fig. 4. Exact queries (number of answers/time)

Dataset	Q'_1	Q'_2	Q'_3	Q'_4
D_1	840/0.015s	605/0.421s	743/0.924s	348/60.7s
D_2	2326/0.055s	605/1s	756/3.17s	925/451s
D_3	3706/0.105s	605/1.6s	767/4.44s	1461/1492s

Fig. 5. Approx/relax queries (number of answers/time)

We see that response times are good for Q'_1 - Q'_3 , allowing many more answers to be returned than for Q_1 - Q_3 within a reasonable amount of time. However, response times for Q'_4 are poor and show a large amount of time required to

return relatively few additional answers compared with Q_4 . The rewriting algorithm generates three new queries at distance 1 starting from Q_4 . One of these queries takes almost all the time shown in the figure (57 seconds on D_1 , 435 seconds on D_2 and 1470 seconds on D_3). This query returns students and courses that they did not attend which were taught by their advisor:

```
SELECT ?X ?Z
WHERE {?X ub:advisor/ub:teacherOf ?Z . ?X !ub:takesCourse ?Z}
```

(Since SPARQL does not accept the symbol “ $_$ ” in triple patterns, we exploit the symbol “ $!$ ” instead. The predicate `!ub:takesCourse` is generated as a result of a substitution operation in which `ub:takesCourse` is substituted by “ $_$ ”. It is sufficient to use `!ub:takesCourse` to match every predicate other than `ub:takesCourse` since the latter will already have been matched.) Since the triple pattern `?X !ub:takesCourse ?Z` matches a large number of triples in the dataset and most of these do not match the first triple pattern, the query returns a limited number of answers in a very long time.

An advantage of our query rewriting approach is that existing techniques for SPARQL query optimisation and evaluation can be reused. Even though this initial experimental study is promising, we will investigate optimizing the rewriting algorithm since it can generate a large number of queries. Moreover, we will also investigate optimization techniques to deal with queries such as Q_4 which cannot be evaluated efficiently using a naive approach.

7 Concluding Remarks and Future Work

We have presented in this paper a study of flexible querying for an extended fragment of the SPARQL 1.1 language. We have shown that adding approximation and relaxation operators to this fragment does not increase the complexity of query answering, and that such operators can be useful in finding more answers than would be returned by the exact form of a user’s query.

We have specified the semantics of our extended language, and have described a query evaluation algorithm based on query rewriting that returns results incrementally according to their “distance” from the original query. Our preliminary experimental studies show promising query performance. Regarding the relative ranking of relaxation and approximation, this depends on the user’s query, the data and the user’s knowledge of the data structuring: if the user uses appropriate predicates but they are too specialised, then RELAX may be more useful; if the user omits part of the necessary structure from their query or uses incorrect predicates, then APPROX may be more useful.

Our ongoing work involves a study of query containment in our extended SPARQL language, and how query costs influence this. Through an investigation of query containment we plan to devise optimizations for our rewriting algorithm. For example, it is possible to decrease the number of queries generated by the rewriting algorithm if $Q = Q_1 \text{ AND } Q_2$ and $[[Q_1]]_G \subseteq [[Q_2]]_G$, then $[[Q]]_G = [[Q_1]]_G$. Our ongoing work also comprises a deeper empirical investigation of the performance of our query evaluation algorithm, and of possible

optimizations. Another direction of research is application of our approximation and relaxation operators, query evaluation and query optimization techniques to federated query processing for SPARQL 1.1.

References

- Alkhateeb, F., Baget, J.-F., Euzenat, J.: Extending SPARQL with regular expression patterns (for querying RDF). *Web Semant.* 7(2), 57–73 (2009)
- Bizer, C., Cyganiak, R., Heath, T.: How to publish linked data on the web. Web page (2007) (Revised 2008) (accessed February 22, 2010)
- Chekoli, M.W., Euzenat, J., Genevès, P., Layaïda, N.: PSPARQL Query Containment. Research report, EXMO - INRIA Grenoble Rhône-Alpes / LIG Laboratoire d’Informatique de Grenoble, WAM - INRIA Grenoble Rhône-Alpes / LIG Laboratoire d’Informatique de Grenoble (June 2011)
- De Virgilio, R., Maccioni, A., Torlone, R.: A similarity measure for approximate querying over RDF data. In: Proceedings of the Joint EDBT/ICDT 2013 Workshops, EDBT 2013, pp. 205–213. ACM, New York (2013)
- Hogan, A., Mellotte, M., Powell, G., Stampouli, D.: Towards fuzzy query-relaxation for RDF. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 687–702. Springer, Heidelberg (2012)
- Huang, H., Liu, C.: Query relaxation for star queries on RDF. In: Chen, L., Triantafillou, P., Suel, T. (eds.) WISE 2010. LNCS, vol. 6488, pp. 376–389. Springer, Heidelberg (2010)
- Huang, H., Liu, C., Zhou, X.: Computing relaxed answers on RDF databases. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 163–175. Springer, Heidelberg (2008)
- Kiefer, C., Bernstein, A., Stocker, M.: The fundamentals of iSPARQL: A virtual triple approach for similarity-based semantic web tasks. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 295–309. Springer, Heidelberg (2007)
- Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006)
- Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. *ACM Trans. Database Syst.* 34(3), 16:1–16:45 (2009)
- Poulouvassilis, A., Wood, P.T.: Combining approximation and relaxation in semantic web path queries. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 631–646. Springer, Heidelberg (2010)
- Reddy, B.R.K., Kumar, P.S.: Efficient approximate SPARQL querying of web of linked data. In: Bobillo, F., Carvalho, R.N., da Costa, P.C.G., d’Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Martin, T., Nickles, M., Pool, M. (eds.) URSW. CEUR Workshop Proceedings, vol. 654, pp. 37–48. CEUR-WS.org (2010)
- Schmidt, M.: Foundations of SPARQL Query Optimization. PhD thesis, Albert-Ludwigs-Universität Freiburg (2009)

How Good Is Your SPARQL Endpoint?[†]

A QoS-Aware SPARQL Endpoint Monitoring and Data Source Selection Mechanism for Federated SPARQL Queries

Muhammad Intizar Ali and Alessandra Mileo

Insight Centre for Data Analytics,
National University of Ireland, Galway, Ireland
`ali.intizar@deri.org`,
`alessandra.mileo@deri.org`

Abstract. Due to the decentralised and autonomous architecture of the Web of Data, data replication and local deployment of SPARQL endpoints is inevitable. Nowadays, it is common to have multiple copies of the same dataset accessible by various SPARQL endpoints, thus leading to the problem of selecting optimal data source for a user query based on data properties and requirements of the user or the application. Quality of Service (QoS) parameters can play a pivotal role for the selection of optimal data sources according to the user's requirements. QoS parameters have been widely studied in the context of web service selection. However, to the best of our knowledge, the potential of associating QoS parameters to SPARQL endpoints for optimal data source selection has not been investigated.

In this paper, we define various QoS parameters associated with the SPARQL endpoints and represent a semantic model for QoS parameters and their evaluation. We present a monitoring service for the SPARQL endpoint which automatically evaluates the QoS metrics of any given SPARQL endpoint. We demonstrate the utility of our monitoring service by implementing an extension of the SPARQL query language, which caters for user requirements based on QoS parameters and selects the optimal data source for a particular user query over federated sources.

1 Introduction

The popularity of semantic technologies is leading to a continuously growing amount of data available on the Web, referred to as Linked Data or the Web of Data. Nowadays, there exist many public SPARQL endpoints providing free access to various linked datasets, which makes it more likely that multiple data sources can qualify to answer a certain query. For example, there are 491

[†] This research has been partially supported by Science Foundation Ireland (SFI) under grant No. SFI/12/RC/2289 and EU FP7 CityPulse Project under grant No.603095. <http://www.ict-citypulse.eu>

SPARQL endpoints listed at DataHub¹, while LODstats² also verifies the presence of over 450 SPARQL endpoints on the Web. A simple keyword based search for *dbpedia*³ on DataHub provides a list of 80 different datasets. The availability of such a large number of data sources and related endpoints is expected to increase the amount of information we can have access to, but it is also increasing the complexity of optimisation and execution of SPARQL queries in a federated infrastructure.

In such a scenario, selecting the optimal data source to answer a certain query is the core ingredient for efficient query processing over federated data sources. Ideally, any increase in number of data sources which can answer a certain triple pattern should not only contribute towards the completeness of the result but can also contribute to efficient federated query processing. However, most of the existing federated SPARQL engines locate relevant datasources either by using precomputed indexes [13,7,1] or by using brute force mechanism (SPARQL ASK) on the fly [17]. As a result, the availability of each new candidate dataset adds extra processing for the query engine, which can have serious impact on the overall performance. This gets more and more crucial in dynamic environments where timely response is key. Monitoring quality metrics of SPARQL endpoints and calculating a ranking of the SPARQL endpoints based on the evaluated quality metrics can drastically reduce the computation cost of federated queries over multiple SPARQL endpoints. Moreover any such quality metrics can facilitate users to choose the best data source based on their preferences.

Several characteristics of the SPARQL endpoints and datasets have been considered for optimal data source selection. Freshness of the data is considered in [19], where authors investigated the tradeoff between fresh and fast processing of the query answers. Recently, data replication has been considered as another important factor for the selection of data sources in federated query processing [15,9]. However, the characteristics of the SPARQL endpoints are not limited to data freshness or replication. Several other factors like quality and licensing of the data as well as performance and availability of SPARQL endpoints etc., are also major concerns for data consumers. Hence, another important factor for selecting the optimal data source when multiple datasets qualify to answer a certain query is which source is best with respect to user's or application's requirements. For example, decision support applications are more concerned about the highest quality of the data while real-time adaptive applications desire to achieve higher performance. However, all of the above mentioned solutions leave the decision of selecting the optimal data source at the discretion of the query engine without taking application-specific requirements into account.

In this paper, we propose a QoS-aware data source selection mechanism for federated SPARQL queries, which takes user's QoS related requirements into account while selecting the optimal data source for query execution. We present a semantic data model for QoS parameters description and calculation by using

¹ A catalogue of Linked data collections. <http://datahub.io> (l.a. May, 2014.)

² <http://stats.lod.eu> (l.a. May, 2014.)

³ A linked data version of the wikipedia: www.dbpedia.org

two popular SPARQL endpoint description vocabularies, namely (i) Vocabulary of Interlinked Datasets (VoID) [2] and, (ii) SPARQL Service Description (SD)⁴. Our main contributions in this paper can be summarised as:

- ★ We identify relevant QoS parameters associated with SPARQL endpoints and present a semantic model for their representation.
- ★ We devise a monitoring system to calculate the values of the QoS parameters.
- ★ We extend the SPARQL query language to enable its users to provide QoS based requirements.
- ★ We evaluate our system using well known benchmark and show case the reduction in the number of the selected data sources as compared to the state of the art federated query engines while achieving QoS compliance at the same time.

Structure of the Paper: We lay the foundations of our study by identifying various QoS parameters associated with SPARQL endpoints and define a semantic model to represent them in Section 2. We present QoS parameters calculation model in Section 3. We demonstrate the utility of our QoS monitoring service for the problem of data source selection and present the conceptual architecture of our proposed solution in Section 4. Section 5 presents our extension of SPARQL query language to cater for QoS requirements. We evaluated our QoS-aware datasource selection algorithm by comparing its performance with the state of the art federated SPARQL query engines in Section 6. We discuss the related work in Section 7 before concluding in Section 8.

2 QoS Parameters for SPARQL Endpoints

In this section we identify various QoS parameters associated with the SPARQL endpoints and categorise them. Later in this section, we design a QoS ontology to represent QoS parameters for SPARQL endpoints.

2.1 QoS Parameters: Definition and Categorisation

QoS is defined in ISO 8042 as: "The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [14]. QoS parameters can play a pivotal role for optimal data source selection in the federated query processing, especially when a set of relevant SPARQL endpoints capable of answering a user's query or triple pattern have been discovered. Table 1 gives an overview of the categorisation and brief description of the QoS parameters associated with the SPARQL endpoints. We intend to give the reader an overview of the QoS parameters which can possibly contribute towards optimal data source selection, hence, this is not intended to be an exhaustive list. We briefly describe each of the QoS categories and their relevant parameters below:

⁴ <http://www.w3.org/TR/sparql11-service-description/>

Table 1. Quality of Service Parameters for SPARQL Endpoints

QoS Category	QoS Parameter	QoS Category	QoS Parameter
Performance	Response Time	Data Quality	Accuracy
	Execution Time		Data Consistency
	Throughput		Completeness
	Error Rate		Freshness
Interoperability	SPARQL Version	Availability	UpTime
	Additional Features		DownTime
	Restricted Features		MeanUpTime
Dataset Description	VoID	Licensing	MTTR
	Service Description		PDDL
ResultSet	SizeLimit		ODC-By
	ResultFormat		ODC-ODbL
			CC0 1.0 Universal

Performance: Performance is the ability to generate the desired outcomes by utilising the appropriate amount of resources and time. Performance of the SPARQL endpoints is determined by accumulative values of the various performance parameters including (i)*response time*: the amount of time required for a SPARQL endpoint to respond a query once received, (ii)*execution time*: amount of time required to execute a certain query over a dataset, execution time is calculated as the amount of time required between the query has been posted by the user and the results are being produced, (iii)*throughput*: number of queries per second a SPARQL endpoint can handle, and (iv)*error rate*: rate at which a certain SPARQL endpoint returns some exceptions or error messages in response to the user queries.

Data Quality: Quality of the data provided by SPARQL endpoints can be judged by evaluating certain data oriented features, like (i)*accuracy*: the degree to which delivered information is correct, (ii)*data consistency*: the degree to which datasets are free from contradictions, (iii)*completeness*: represents that the result set provided by SPARQL endpoint is of sufficient size or number of triples returned for the particular query, and (iv)*data freshness*: refers to the age of the data. It is the degree to which extent data provided by the endpoint is up-to-date.

Interoperability Interoperability among SPARQL endpoints is their ability to answer certain query language specific features. Most of the public SPARQL endpoints has capacity to process SPARQL 1.0⁵, while there are some early adaptors of the SPARQL 1.1⁶ as well [5]. *SPARQL version*: reflects that which version of the SPARQL query language is supported by the SPARQL endpoint. *Restricted features*: provide a list of restricted features of a query language not supported by the SPARQL endpoint. It is common practice to restrict some features of the underlying query language to avoid work load or network traffic e.g. SPARQL ASK queries. *Additional features*: provides

⁵ <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>

⁶ <http://www.w3.org/TR/sparql11-query/>

a list of the additional features supported by any SPARQL endpoints as a result of the extension of the query language.

Availability: Ability of the SPARQL endpoint to answer an authorised user's query at any given time [14]. Availability is very critical in the distributed and autonomous architecture where users or applications lack the control over the underlying query engines. Availability related QoS parameters of SPARQL endpoint are, (i) *uptime*: denotes the date and time when the SPARQL endpoint was made available for access, (ii) *downtime*: refers to the time and date since when a SPARQL endpoint is offline, (iii) *mean up time*: is the average amount of time a SPARQL endpoint remains available for authorised users within a certain time frame, and (iv) *mean time to recover (MTTR)*: is the average amount of time a SPARQL endpoint takes to recover if it goes offline due to some technical or operational reasons.

Dataset Description: Ability of the SPARQL endpoints to provide meta information about the datasets hosted using data description vocabularies. Data description vocabularies are also used by application and query engines to automatically discover relevant datasets. Two popular methods for SPARQL endpoint dataset description are VoID and SD.

ResultSet: Data oriented nature of the SPARQL endpoints brings some additional QoS parameters relevant to the result set generated by the endpoint while giving answer to a certain query. *Size limit* is mostly imposed by SPARQL endpoints to limit the maximum number of triples which can be returned by a SPARQL endpoint while answering a certain query. Limit is imposed to reduce the network traffic caused by queries having reasonably larger result size. Another important factor of user's concern about the result set generated by SPARQL endpoint is the format of the results.

Licensing: Licensing defines the liabilities, legal requirements and usage permission for the data sets. Several licensing agreements are in practice, VoID describes four popular data licensing agreements used by SPARQL endpoints. We also limit ourselves to these four types of licenses which are, (i) Public Domain Dedication and License (PDDL), (ii) Open Data Commons Attribution (ODC-By), (iii) Open Database License (ODC-ODbL), and (iv) Creative Commons Public Domain Waiver (CC0 1.0 Universal).

2.2 QoS Parameters: Semantic Description Model

Inspired by the QoS ontology for Web services described in [20], we defined a QoS ontology for quality related parameters of SPARQL endpoints. We partially reuse existing vocabularies such as VoID, SD and QoS Ontology [20,8]. We define a *QoS Profile* class which can be attached to the *void:dataset*⁷ or *sd:service*⁸ class using *qs:hasQoSProfile*⁹. Each QoS parameter belongs to a specific category, which is a subclass of QoS. We use some existing properties of VoID and

⁷ Prefix void: <<http://rdfs.org/ns/void#>>

⁸ Prefix sd: <<http://www.w3.org/ns/sparql-service-description#>>

⁹ Prefix qs: <<http://www.deri.org/QoS#>>

SD for the QoS parameters. For example, QoS parameter *ResultFormat* as defined previously, can be easily described by using *sd:resultFormat* property of SD. Similarly, we use *dcterms:license* from VoID to describe QoS parameter *Licensing*. We use *Measurement Units Ontology*¹⁰ to measure the values of the QoS parameters. We limited ourselves to the domain independent generic QoS parameters, however any additional domain specific or domain independent QoS parameters can be easily augmented. It is also worth mentioning that we only provided the definition and categorisation of the subset of the QoS parameters while defining and categorising a complete set of all possible QoS parameters is out of the scope of this paper.

Definition 1. Let $P_{qos} = \{qos_i \mid i = 1..s\}$ be the QoS profile of a SPARQL endpoint, representing the set of all QoS parameters associated to that endpoint. We define P_{qos}^U as the set of user's defined QoS attributes and P_{qos}^D as the set of default QoS parameters indicated by the system, such that $P_{qos}^U \cup P_{qos}^D \subseteq P_{qos}$.

Furthermore, we define $PQ_{qos} = \{qos_k \mid k = 1..t\}$ as a QoS profile for a query Q such that $PQ_{qos} = P_{qos}^U \cup P_{qos}^D$ and $PQ_{qos} \subseteq P_{qos}$.

Most of time, a user doesn't want to specify (or is not aware of) all of the QoS parameters associated to a SPARQL endpoint, therefore we introduce the concept of the P_{qos}^D which describes some critical QoS parameters and their minimum threshold expected from any SPARQL endpoint listed by the federated engine. We assume this is set by the federated query engine in order to ensure that a reasonable QoS standard is maintained even without any explicit requirements, i.e. when $P_{qos}^U = \emptyset$. In this case, $PQ_{qos} = P_{qos}^D$. Listing 1 shows a sample QoS description of a SPARQL endpoint using our QoS semantic model.

Listing 1. A Sample QoS Profile of a SPARQL Endpoint

```

<http://www.deri-testbed/QoS/endpoint1> a sd:Service ;
sd:endpointUrl <http://deri.org/QoS/sparql>;
sd:supportedLanguage sd:SPARQL11Query;
sd:resultsFormat <http://www.w3.org/ns/formats/RDF_XML> ;
dcterms:license <http://www.opendatacommons.org/licenses/pddl/> ;

qs:hasQoSProfile qs:QoSProfileEndpoint120140514T105940 .

qs:QoSProfileEndpoint120140514T105940
qs:isGeneratedAt "2014-05-14T10:59:40Z" ;
qs:contains qs:QoSProfileEndpointParameter20140514T105940Completeness.

qs:QoSProfileEndpointParameter20140514T105940Completeness
qs:hasName "Completeness" ;
qs:hasTendency "increase" ;
qs:isMeasuredIn muo:UnitOfMeasurement ;
qs:metricType qs:NumericMetric ;
qs:hasValue "70" .

```

¹⁰ Prefix muo: <http://purl.oclc.org/NET/muo/muo#UnitOfMeasurement>

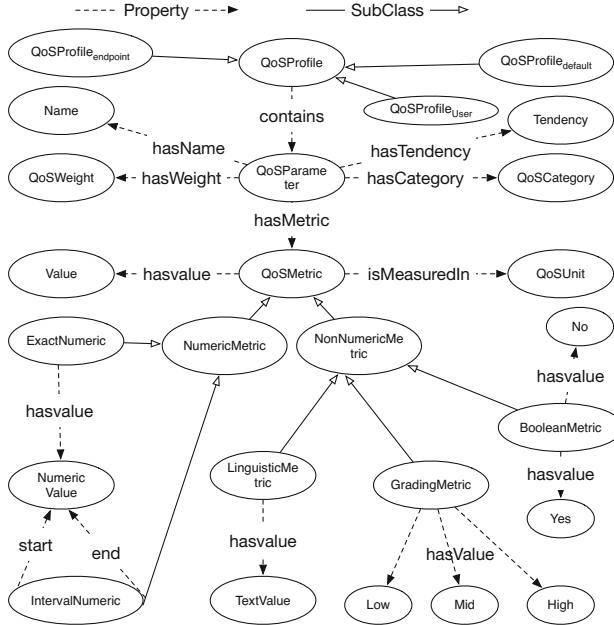


Fig. 1. A QoS Parameters Evaluation Metrics Ontology [8]

3 Evaluation of QoS Parameters

In this section, we present a QoS parameters calculation ontology by defining QoS metrics and their calculation method.

3.1 QoS Parameters: Evaluation Metrics Semantic Model

We present an extensible semantic model for the QoS parameters evaluation of any corresponding SPARQL endpoint. Our presented semantic data model can be easily extended to cater for new QoS parameters or categories and their evaluation mechanism. QoS ontology for SPARQL endpoints evaluation metrics is designed by closely examining the VoID and SD as well as existing QoS specifications for web services [20]. Figure 1 depicts fundamental concepts of QoS parameters and their evaluation metrics. Top level concept is *QoSProfile* as defined in the previous section which contains three subclasses namely, (i) *QoSProfileUser* reflecting user's defined QoS requirements, (ii) *QoSProfileDefault* represents default values of the QoS requirements configured within the system, and (iii) *QoSProfileEndpoint* is the QoS summary of a SPARQL endpoint.

QoS Parameter indicates the quality attribute of the *QoS Profile* while *QoS Metric* defines the measurement of the *QoS Parameter*. *QoS Metric* can be either, (i) *Numeric*: where value of a QoS parameter can be measured as an exact

numeric value or an interval between two numeric values , or (ii) *NonNumeric*: whose value can be boolean (true, false), grades (high, middle, low), or in linguistics as any text literal e.g. SPARQL1.1 is a linguistic value for the *SPARQL Version* parameter. *QoS Tendency* indicates the effects of the change in the value of QoS parameter on the aggregated QoS. Tendency can be mentioned as either increase or decrease, where increase denotes that any increase in the QoS parameter value is directly proportional to the QoS value (e.g. increase in the accuracy level results into higher data quality), while decrease denotes that any increase in the QoS parameter value is inversely proportional to the QoS value (e.g. increase in execution time results into lower performance). Tendency is mostly relevant for the QoS parameters containing numeric values.

3.2 QoS Parameters: Monitoring and Value Calculation

Measurement of the value of the QoS is a challenging task because of multidimensional nature of the QoS parameters and their tendency. We used simple evaluation method by executing monitoring queries with the mix of both active approach (continuous monitoring of SPARQL endpoints by periodic sample and testing) and passive approach (runtime QoS evaluation at the query arrival). Below we discuss methods we used to calculate the value of the each of the QoS parameter discussed in Section 2.

Performance: Response time of a SPARQL endpoint can be measured by executing simple ASK queries. However, to avoid network congestion and malicious attacks many SPARQL endpoints restrict ASK queries. We use a below mentioned simple SPARQL SELECT query where $< s >$ and $< o >$ are randomly generated URI to make sure that query result set is empty.

```
Q1.      SELECT ?p where { <s> ?p <o> }
```

However to avoid the influence of any indexes maintained by SPARQL endpoint over the performance evaluation, we used some additional *Subject-Subject*, *Object-Object* and *Subject-Object* join queries as described in [6]. Q2 is a sample *Object-Object* join query.

```
Q2.      SELECT ?o where { s1 p1 ?o  
                      s2 p2 ?o }
```

We estimated query execution time by calculating the time required to get the answers of the Q1, Q2 and Q3 respectively and subtracted their response time. Q2 can also be further extended to formulate complex queries while covering more Basic Graph Patterns (BGPs). We executed the query with varying result size by imposing LIMIT of 10, 100 and 1000 respectively.

```
Q3.      SELECT * where { ?s ?p ?o } LIMIT 1000
```

Throughput was observed by repeatedly sending Q1 and error rate was measured by counting the number of exceptions returned by the SPARQL endpoint while answering all executed queries.

Data Quality: Measurement of data quality is not an easy task because multiple factors can contribute to the data quality. We use only simplified basic methods to calculate completeness and freshness of the data. Data completeness of a SPARQL endpoint is checked by comparing the result size (number of triples) generated by a SPARQL endpoint over multiple executions. We used VoID property *dcterms:modified* to check the freshness of data. We have to rely on *dcterms:created* in cases where *dcterms:modified* was non-existent. We left the application of any sophisticated methods for the evaluation of accuracy and consistency of the result set and to find out any incorrectness, inconsistency and/or contradictions in the data as part of our future work.

Interoperability: *SPARQL Version* parameter indicates which version of the language is supported by the endpoint. We used SPARQL1.1 test data set¹¹, which provides complete set of tests to check the compliance of the SAPRQL endpoint with SPARQL 1.1. Similarly, restricted features can be listed by evaluating the same test queries to evaluate supported features of the language. We rely on the endpoint service provider for the provision of the initial list of features if any additional features are supported, however additional features can be validated using the same approach as used for restricted features evaluation.

Availability: Initial value for the *UpTime* of a SPARQL endpoint is obtained through *dcterms:date* property of the VoID and can be periodically monitored by executing Q1. The value of *DownTime* is achieved by continuously monitoring the SPARQL endpoint with Q1 till any non-response or exception in the execution of Q1 is reported. *Mean UpTime* and *MTTR* is calculated after analysing the historical data of observations of *UpTime* and *DownTime*. We monitored our testbed endpoints only for a period of 90 days, however Linked Open Data Analytics initiatives like SPARQLES can play an important role for the evaluation of these metrics [5].

Dataset Description: SPARQL endpoints are mostly described by VoID or SD. Service description is retrieved by dereferencing the endpoint URI, while VoID description is obtained by executing Q4.

```
Q4.      PREFIX void: <http://rdfs.org/ns/void#>
          SELECT ?ds WHERE {
            ?ds a void:Dataset .
            ?ds void:SPARQLEndpoint <SPARQLEnpointURI> }
```

ResultSet: Most of the SPARQL endpoints limit the size of result set to reduce the network traffic. Variation of Q3 with varying limit size is executed to monitor if there is any limit imposed on the size of the result set. A query over SD using Q5 can provide a list of supported formats by any SPARQL endpoint where <endpointURI> corresponds to the URI of the relevant SPARQL endpoint.

¹¹ <http://www.w3.org/2009/sparql/docs/tests/data-sparql11/>

```

Q5.
PREFIX sd: <http://www.w3.org/ns/sparql-service-description#>
SELECT ?format WHERE {
    ?s a sd:service .
    ?s sd:endpoint <endpointURI> .
    ?s sd:resultFormat ?format . }

```

Licensing: VoID provides four basic licensing modes for the licensing of the datasets. Q6 returns the particular license type supported by the SPARQL endpoint.

```

Q6.      PREFIX dcterms: <http://purl.org/dc/terms/>
SELECT ?license WHERE {
    ?ds a void:Dataset .
    ?ds dcterms:license ?license . }

```

4 QoS-Aware Data Source Selection

In this section, we define the problem of QoS-aware data source selection and showcase the utility of our QoS monitoring service by presenting the conceptual architecture of our proposed solution.

4.1 Problem Statement

Data source selection is the process of determining which candidate data source will be selected if there are multiple sources available to answer a certain query. The general problem of data source selection is as in Definition 2.

Definition 2. *Given a user's query Q and set of n data sources $DS = \{ds_i \mid i = 1..n\}$, we define $DS_c = \{ds_j \mid j = 1..m\}$ as a set of candidate datasources that can potentially contribute to answer query Q , where $DS_c \subseteq DS$ and $1 \leq m \leq n$.*

QoS-Aware data source selection as described in Definition 3 is the process of determining which candidate data sources will contribute to answer a certain query Q in such a way to be compliant with the QoS requirements specified by the user or the application.

Definition 3. *Given a user's query Q and set of candidate data sources DS_c , we define $DS_{qos} = \{ ds_{qos} \mid k = 1..l \}$ as the set of optimal data sources that can potentially contribute to the answer query Q and are compliant with the QoS requirements mentioned in the query, where $DS_{qos} \subseteq DS_c$ and $1 \leq l \leq m \leq n$.*

4.2 Conceptual Architecture

Figure 2 depicts the conceptual architecture of our proposed QoS-aware federated SPARQL query processing system. A user's query with QoS requirements is processed by our extended *QoS-aware Query parser* which generates QoS profile of the query and sends it (without QoS requirements) to the federated query engine. We left at the discretion of the federated query engine to find out multiple candidate sources containing similar data for each triple pattern. Mostly

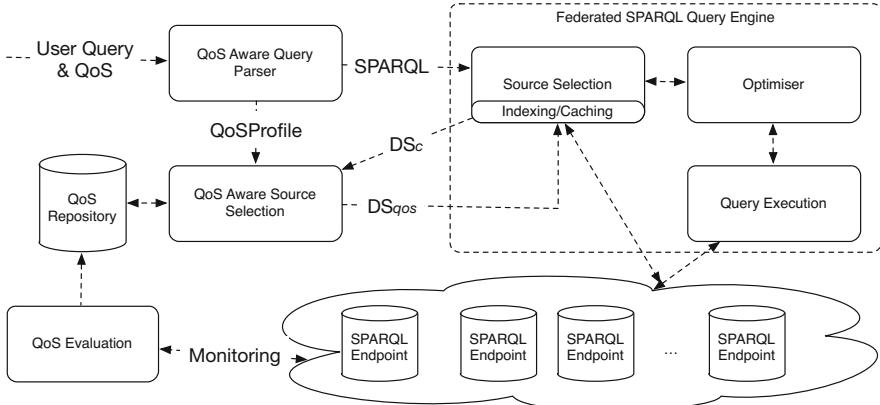


Fig. 2. QoS-aware Federated SPARQL Query Processing

existing federated query engines calculate triple pattern wise candidate sources either using precomputed indexes or evaluating it at run time by sending ASK queries. The *QoS-Aware Source Selection* component receives triple pattern wise candidate data sources (DS_c) from the source selection component of the federated query engine and returns a reduced number of data sources (DS_{qos}) after evaluating their QoS compliance with QoS requirements specified in the query. QoS compliance is verified using the QoS repository generated by the *QoS Evaluation* component which continuously monitors QoS parameters of all SPARQL endpoints accessible by the federated query engine.

5 QoS Aware SPARQL Query Extension

In this section we introduce our QoS-aware extension of the SPARQL query language and describe our algorithm for optimal QoS-aware data source selection.

5.1 Syntax and Semantics

Our aim for the extension of the SPARQL query language is to enhance its ability to cater for user requirements in terms of QoS parameters. We introduced a new keyword *QOSREQ* by extending the grammar of SPARQL 1.1 extension for federated queries. The set of query related QoS requirements qos_k as per Definition 1, are attached to each *triple pattern*. User can apply *QOSREQ* operator either at BGP level and/or triple pattern level. In the former case, if a user defines QoS constraints at Basic Graph Pattern (*BGP*) level, all the QoS constraints are equally assigned to each of the triple patterns within the *BGP* accordingly. However, in the later case, if user describes QoS requirements for few of the individual triple patterns, remaining triple patterns will be assigned default values

of the QoS requirements using default QoS profile. In cases where QoS requirements are defined for both BGP and triple pattern, QoS requirements defined at triple pattern level will supersede the QoS requirements defined at BGP level. *QOSREQ* operator attaches the QoS requirements to immediately preceding triple pattern or BGP .

Definition 4. Let us consider the set of query-related QoS parameters PQ_{qos} as per Definition 1. We define set of Qos requirements

$$R_{qos} = \{r_k = \langle qos_k \ op \ val_k \rangle \mid qos_k \in PQ_{qos}, \ op \in \{<, >, =\}\}$$

where val_k is the value constrained by operator op to qos_k .

Definition 5. Let us consider the set of QoS parameters P_{qos} as per Definition 1, related to a datasource $dsc_j \in DS_c$. We define the set of QoS configuration for a candidate datasource as

$$C_{qos}(dsc_j) = \{c_i(dsc_j) = \langle qos_i \ setval_i \rangle \mid qos_i \in P_{qos}, \ dsc_j \in DS_c\}$$

where $setval_i$ is the value of qos_i for the candidate datasource dsc_j .

Listing 2. A Sample QoS-Aware SPARQL Query

```

SELECT ?drug ?keggUrl ?chebiImage
WHERE {
  ?drug rdf:type drugbank:drugs .
  QOSREQ[qs:Completeness = 80, qs:SizeLimit > 10000]

  ?drug drugbank:keggCompoundId ?keggDrug .
  ?keggDrug bio2rdf:url ?keggUrl .

  { ?drug drugbank:genericName ?drugBankName .
    ?chebiDrug purl:title ?drugBankName . }
  QOSREQ[qs:DatasetDescription = 'VoID',
  qs:ResponseTime < 30 ]

  ?chebiDrug chebi:image ?chebiImage . }
```

Note that in the current implementation we consider all set of operators (e.g. $op \in \{<, >, =\}$) and their combinations (e.g. \leq, \geq) for qos_k with numeric and grading values, while for qos_k ranging over string values, only equality is considered as a valid operator ($op \in \{=\}$), e.g. $\langle license = "PDDL" \rangle$.

If a user query explicitly defines the endpoint using the *SERVICE* keyword, this is given priority over the QoS constraints (which are all discarded) and the SPARQL 1.1 query is directly fed into the federated query engine supporting the language. We do not show extension of the SPARQL 1.1 grammar rules for the introduction of *QOSREQ* operator due to space limitation. Listing 2 shows a sample QoS-aware SPARQL query over federated sources. Prefixes are ignored due to space limitations.

5.2 Source Selection

Algorithm 1 shows the QoS-aware optimal data source selection process. Input for the algorithm are (i) the user query with explicit definition of the requirements $r_k \in R_{qos}$ attached to the query or to a triple pattern in the query and (ii) the set of candidates data sources $DS_c(tp_t)$ for each triple pattern tp_t . The output of the algorithm is a reduced subset DS_{qos} of the candidate services DS_c which comply with the QoS requirements in R_{qos} . In case multiple sources can fulfil the set of QoS requirements for the query or for a BGP, we let the underlying federated query engine to make the final decision, however it can also be prioritise on the overall score of the QoS parameters or reputation of the SPARQL endpoint. Similarly, if none of the data sources fulfil the QoS criteria specified in the query, we let federated query engine to opt for either returning empty result set or compromising over QoS criteria. We limit ourselves to the individual values of the QoS parameters, we refer our readers to the extensive literature on methods of calculating the overall QoS score of a service by aggregation of all the QoS parameters while considering their weight-age, tendency and preferences [8,11]. Let's identify the triple patterns in the query as $TP = \{tp_1, tp_2, \dots, tp_t\}$ and parametrise the set of requirements R_{qos} with the pattern they refer to as $R_{qos}(tp_t) = \{r_k(tp_t) \mid r_k \in R_{qos}, tp_t \in TP\}$. We do the same for the candidate sources DS_c and for the optimal sources DS_{qos} obtaining $DS_c(tp_t) = \{dsc_i(tp_t) \mid dsc_i \in DS_c, tp_t \in TP\}$ and $DS_{qos}(tp_t) = \{dsqos_j(tp_t) \mid dsqos_j \in DS_{qos}, tp_t \in TP\}$. The process is iterated for each tp_t in the query as shown in Algorithm 1.

```

input :  $R_{qos}(tp_t)$  ,  $DS_c(tp_t)$ 
output:  $DS_{qos}(tp_t)$ 

foreach  $tp_t \in TP$  do
     $DS_{qos}(tp_t) = \emptyset$ 
    foreach  $dsc_i \in DS_c(tp_t)$  do
        flag = true;
        foreach  $r_k(tp_t) = \langle qos_k op val_k \rangle \in R_{qos}(tp_t)$  do
            if  $setval_k op val_k$  when  $c_k(dsc_i) = \langle qos_k setval_k \rangle$  then
                do nothing;
            else
                | flag = false;
            end
        end
        if  $flag == true$  then
            |  $DS_{qos}(tp_t) \leftarrow DS_{qos}(tp_t) \cup dsc_i(tp_t)$ 
        end
    end
    return  $DS_{qos}(tp_t)$ 
end

```

Algorithm 1. QoS-Aware Data Source Selection Algorithm

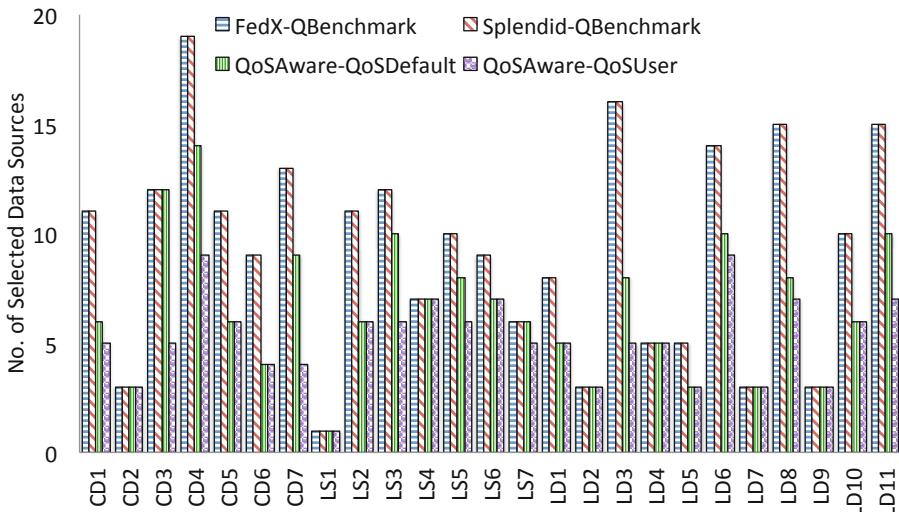


Fig. 3. Comparison of the Number of Data Sources Selected (Federated Engines)

6 Experimental Evaluation

Implementation & Testbed: We extended SPARQL 1.1. grammar to introduce *QOSREQ* operator. Implementation was carried out using Java 1.7 and antlr¹² was used for java code generation from SPARQL grammar rules. We used Fedbench [16] benchmark for the simulation of the federated queries and deployed the benchmark datasets over 9 SPARQL endpoints¹³. All the endpoints were deployed over Virtuoso¹⁴. We created one replica of each dataset and hosted it on different endpoint with varying configuration. This variation helped us for the generation of different QoS parameter's values of two SPARQL endpoints hosting same datasets. We opted for a controlled environment for the initial evaluation because of the availability of the evaluation results of the existing federated query engines using FedBench dataset and queries. However, our monitoring service can be easily deployed over public SPARQL endpoints.

Monitoring QoS Parameters: QoS parameters summaries were maintained and indexed for all deployed SPARQL endpoints. We used QoS parameters defined in Table 1 as an initial set of QoS parameters. We monitored each of the SPARQL endpoint over a period of 90 days to observe any changes in QoS parameters values and updated them accordingly. We maintained a QoS repository

¹² <http://www.antlr.org>

¹³ Authors are thankful to Muhammad Saleem, AKSW, University of Leipzig for the provision of FedBench datasets. Detailed description of the SPARQL endpoints can be found at <http://deri-srvgal29.nuig.ie:8080/QoS>

¹⁴ <http://virtuoso.openlinksw.com>

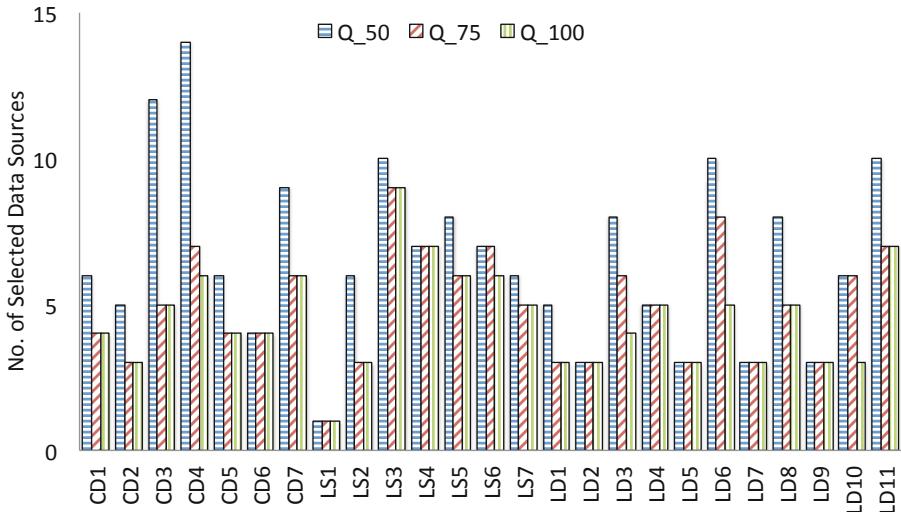


Fig. 4. Comparison of the Number of Data Sources Selected (QoS Variations)

of all the SPARQL endpoints of the testbed¹⁵. Regular monitoring was set at every 24 hours with equal distribution between 12:00 a.m and p.m. to reduce the effect of network traffic during busy hours. Predictably, there were not so much frequency of changes in the values of the QoS parameters because experimental evaluation was conducted in a controlled local network. In order to increase the frequency of the fluctuation in the QoS parameters values some technical hazards and endpoint data policies were changed with human interaction at random intervals e.g. stopping virtuoso server to reflect unavailability or updates in one dataset while keeping the replica outdated. We opted for the in-memory storage of QoS parameters repository due to its smaller size.

Evaluation Results: We ran 25 queries provided with Fedbench over the testbed. We counted total number of triple pattern wise selected sources for each query. We used SPLENDID (*index based*)[7] and FedX(*index free*)[17] query engines for the execution of the federated queries and demonstrate compatibility of our extension with both types of SPARQL engines. We ran three versions of the each of the benchmark query with (i)*Q_{Benchmark}*: containing Fedbench queries without any modification, (ii) *QoS_{default}*:values of the QoS parameters defined within QoS default profile were attached to the benchmark queries, and (iii) *QoS_{User}*: we assigned QoS requirements randomly to each of the benchmark query ¹⁶. We ran 10 execution of each query and counted the triple pattern wise

¹⁵ A sample QoS repository is available at: <http://deri-srvgal29.nuig.ie:8080/QoS>

¹⁶ Detailed description of the QoS Repository, *Q_{Benchmark}*, *QoS_{default}*, *QoS_{User}*, *Q₅₀*, *Q₇₅*, and *Q₁₀₀* are accessible at: <http://deri-srvgal29.nuig.ie:8080/QoS/>

average total number of data sources selected for each query. Figure 3 shows the decrease in the number of the *tp-wise* selected sources while executing *QoS-Aware* queries over Fedbench as compared to the number of data sources selected by the existing federated query engines. $Q_{Benchmark}$ queries were executed using FedX and SPLENDID, while $QoS_{default}$ and QoS_{User} were executed using our QoS-Aware source selection algorithm. QoS Aware source selection algorithm considers candidate data sources returned by SPLENDID. If QoS constraints are defined, our system compares QoS profile of the query with the QoS profile of the relevant data sources, and returns the reduced number of the data sources. We verified 100% compliance of the QoS requirements for each of the selected data source by our system while executing $QoS_{default}$ and QoS_{User} . In order to observe the effect of increase and decrease in the threshold of the QoS parameters, we devised three versions of the each of benchmark query with different threshold levels, (i) Q_{50} , (ii) Q_{75} , and (iii) Q_{100} , where values of the QoS parameters were set at <50%, between 50 -75% and over 75% of the best possible value for a parameter respectively, while taking tendency of the QoS parameters into account. Three possible values for the QoS *GradingMetrics*, LOW, MED, HIGH were assigned within Q_{50} , Q_{75} and Q_{100} respectively. It was interesting to observe that most of the QoS parameters fall into the category of either Q_{50} or Q_{100} mostly because of the boolean nature of the majority of the selected QoS parameters. Figure 4 shows the decrease in the number of selected data sources for Q_{50} and Q_{75} as compared to Q_{100} .

7 Related Work

Federated query processing over distributed SPARQL endpoints have been widely studied and different approaches to process federated queries and their optimisation have been proposed [17,7,1,13]. Federated SPARQL engines can be classified into two categories (i) *Index based*, and (ii) *Index free*. Independently of the use of an index, in both approaches an initial list of all SPARQL endpoints URI has to be provided to initialise the query engine [17]. This might result in limited flexibility to cater for new sources, but also produce an increased number of potential sources providing the required data. Recently, data replication has been considered to reduce the number of potential data sources which can answer a user's query [9,15]. Data source selection based on the tradeoff between data freshness and fast query processing is presented in [19]. However, broader categories of the QoS parameters associated with SPARQL endpoints for more effective data and application driven source selection has not been explored yet.

QoS parameters have been extensively studied in the context of Web services [20]. Initial work is focused on identification and definition of QoS parameters, while later work was built on critical aspects of relevant web service selection especially when a set of relevant sources is discovered [10]. A QoS ontology based service selection model is presented in [20]. However, nature of SPARQL endpoints is more inclined towards data services rather than the traditional Web services paradigm. In [18,3,4], authors advocated that data services are more

concerned about quality, ownership and usage permission of the data which makes them different from the traditional Web services. QoS parameters for SPARQL endpoints while considering the data oriented nature of the SPARQL endpoints have not been realised so far.

User QoS related requirements have been traditionally studied in many fields of science including databases [12]. Two prevalent approaches to cater for user's QoS requirements in the databases are (i) *query extension*: existing query languages are extended to express user's QoS requirements within query language, or (ii) *QoS-aware query language*: a new query language is proposed to express user QoS requirements. Compatibility and non-adaptability are two major concerns of the later approach, therefore we opted for the former approach and propose an extension of the SPARQL query language which enables expressivity of the QoS requirements of a particular user within the SPARQL query language.

8 Conclusion and Future Directions

In this paper, we presented a QoS-aware SPARQL endpoint monitoring and data source selection mechanism for SPARQL queries over federated endpoints. We identified various QoS parameters of SPARQL endpoints and presented semantic model to represent and evaluate these parameters. We calculated values of the QoS parameters by continuous monitoring and generating QoS profiles of the SPARQL endpoints. We extended SPARQL query language to enable users to define their QoS requirements. We showcase the effectiveness of our QoS monitoring service by using the data source selection problem of the federated SPARQL queries. Our evaluation results show the reduction in the number of selected data sources as compared to state of the art federated query engines, while achieving QoS compliance according to the user's specified requirements.

In future work, we plan to further explore broader range of QoS parameters and generate a single aggregated QoS value for each SPARQL endpoint particularly for the public SPARQL endpoints. We also plan to integrate user feedback for the fair evaluation of the QoS parameters. Application of more sophisticated methods for the evaluation of Quality of Information (QoI) parameters of the datasets accessible via SPARQL endpoints is also part of our future agenda.

In the context of smart cities applications, we plan to extend our QoS-aware federated query for linked stream data sources, enabling users and application to adaptively select optimal data source based on the changing requirements, preferences and context.

References

1. Acosta, M., Vidal, M.-E., Lampo, T., Castillo, J., Ruckhaus, E.: ANAPSID: An adaptive query processing engine for SPARQL endpoints. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 18–34. Springer, Heidelberg (2011)
2. Alexander, K., Hausenblas, M.: Describing linked datasets, the vocabulary of interlinked datasets. In: Proc. of LDOW, WWW (2009)

3. Ali, M.I., Pichler, R., Truong, H.L., Dustdar, S.: Data concern aware querying for the integration of data services. In: Proc. of ICEIS (1), pp. 111–119 (2011)
4. Ali, M.I., Pichler, R., Truong, H.-L., Dustdar, S.: Incorporating data concerns into query languages for data services. In: Zhang, R., Zhang, J., Zhang, Z., Filipe, J., Cordeiro, J. (eds.) ICEIS 2011. LNBP, vol. 102, pp. 132–145. Springer, Heidelberg (2012)
5. Aranda, C.B., Hogan, A., Umbrich, J., Vandenbussche, P.-Y.: Sparql web-querying infrastructure: Ready for action? In: Proc. of ISWC (2), pp. 277–293 (2013)
6. Gallego, M.A., Fernández, J.D., Martínez-Prieto, M.A., de la Fuente, P.: An empirical study of real-world sparql queries. In: Proc. of USEWOD 2011, at WWW (2011)
7. Görlitz, O., Staab, S.: Splendid: Sparql endpoint federation exploiting void descriptions. In: Proc. of COLD, vol. 782 (2011)
8. Guo, G., Yu, F., Chen, Z., Xie, D.: A method for semantic web service selection based on qos ontology. Journal of Computers 6(2) (2011)
9. Hose, K., Schenkel, R.: Towards benefit-based rdf source selection for sparql queries. In: Proc. of SWIM, p. 2 (2012)
10. Huang, A.F., Lan, C.-W., Yang, S.J.: An optimal qos-based web service selection scheme. Journal of Information Sciences 179(19), 3309–3322 (2009)
11. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benrernou, S., Brandic, I., Kertész, A., Parkin, M., Carro, M.: A survey on service quality description. ACM Computing Surveys (CSUR) 46(1), 1 (2013)
12. Mobedpour, D., Ding, C., Chi, C.-H.: A qos query language for user-centric web service selection. In: Proc. of SCC 2010, pp. 273–280. IEEE (2010)
13. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
14. Rafique, I., Lew, P., Abbasi, M.Q., Li, Z.: Information quality evaluation framework: Extending iso 25012 data quality model. In: Proc. of World Academy of Science, Engineering and Technology, vol. 65 (2012)
15. Saleem, M., Ngonga Ngomo, A.-C., Xavier Parreira, J., Deus, H.F., Hauswirth, M.: DAW: Duplicate-aWare federated query processing over the web of data. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 574–590. Springer, Heidelberg (2013)
16. Schmidt, M., Görlitz, O., Haase, P., Ladwig, G., Schwarte, A., Tran, T.: FedBench: A benchmark suite for federated semantic data query processing. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 585–600. Springer, Heidelberg (2011)
17. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization techniques for federated query processing on linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 601–616. Springer, Heidelberg (2011)
18. Truong, H.L., Dustdar, S.: On evaluating and publishing data concerns for data as a service. In: Proc. of APSCC, pp. 363–370 (2010)
19. Umbrich, J., Karnstedt, M., Hogan, A., Parreira, J.X.: Hybrid SPARQL queries: Fresh vs. Fast results. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 608–624. Springer, Heidelberg (2012)
20. Wang, X., Vitvar, T., Kerrigan, M., Toma, I.: A qos-aware selection model for semantic web services. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 390–401. Springer, Heidelberg (2006)

Embedding OWL Querying and Reasoning into XQuery^{*}

Jesús M. Almendros-Jiménez

Dpto. de Informática Universidad de Almería 04120-Spain
jalmen@ual.es

Abstract. In this paper we present a framework called XQOWL that makes possible to handle XML and RDF/OWL data with XQuery. XQOWL can be considered as an extension of the XQuery language that connects XQuery with SPARQL and OWL reasoners. XQOWL embeds SPARQL queries (via Jena SPARQL engine) into XQuery and enables to make calls to OWL reasoners (HermiT, Pellet and FaCT++) from XQuery. It permits to combine queries against XML and RDF/OWL resources as well as to reason with RDF/OWL data. Therefore input data can be either XML or RDF/OWL and output data can be formatted in XML (also using RDF/OWL XML serialization).

1 Introduction

There are two main formats to publish data on the Web. The first format is *XML*, which is based on a tree-based model and for which the *XPath* and *XQuery* languages for querying, and the *XSLT* language for transformation, have been proposed. The second format is *RDF* which is a graph-based model and for which the *SPARQL* language for querying and transformation has been proposed. Both formats (XML and RDF) can be used for describing data of a certain domain of interest. The *W3C (World Wide Web Consortium)*¹ proposes transformations from XML data to RDF data (called *lifting*), and vice versa (called *lowering*). RDF have XML-based representations (called *serializations*) that makes possible to represent in XML the graph based structure of RDF.

One of the main aims of the so-called Web of Data is to be able to handle heterogeneous resources where data can be expressed in either XML or RDF. The design of programming languages able to handle both XML and RDF data is a key target in this context and some recent proposals have been presented with this end. The main gain in having the integrated language is the performance, the reduction on complexity and the flexibility. One of most known is XSPARQL [5] which is an hybrid language which combines XQuery and SPARQL allowing to query XML and RDF. XSPARQL extends the XQuery syntax with new expressions able to traverse a RDF graph and construct the graph of the result of a

^{*} This work was supported by Spanish MINECO Ministry (*Ministerio de Economía y Competitividad*) under grant TIN2013-44742-C4-4-R, as well as by the Andalusian Regional Government (Spain) under Project P10-TIC-6114.

¹ <http://www.w3.org/>

query on RDF. One of the uses of XSPARQL is the definition of lifting and lowering from XML to RDF and vice versa. The SPARQL2XQuery interoperability framework [4] aims to overcome the same problem by considering as query language SPARQL for both formats (XML and RDF), where SPARQL queries are transformed into XQuery queries by matching XML Schemas into RDF metadata. In early approaches, SPARQL queries are embedded into XQuery and XSLT [7] and XPath expressions are embedded into SPARQL queries [6].

OWL is an ontology language working with concepts (i.e., classes) and roles (i.e., object/data properties) as well as with individuals (i.e., instances) which fill concepts and roles. OWL can be considered as an extension of RDF in which a richer vocabulary allows to express new relationships. OWL reasoning [16] is a topic of research of increasing interest in the literature. Most of OWL reasoners (for instance, *Hermit* [12], *Racer* [9], FaCT++ [17], Pellet [15]) are based on tableau-based decision procedures.

SPARQL provides mechanisms for querying tasks while OWL reasoners are suitable for reasoning tasks. SPARQL is a query language for RDF/OWL triples whose syntax resembles SQL. OWL reasoners implement a complex deduction procedure including ontology consistency checking that SPARQL is not able to carry out. Therefore SPARQL/OWL reasoners are complementary in the world of OWL.

In this paper we present a framework called XQOWL that makes possible to handle XML and RDF/OWL data with XQuery. XQOWL can be considered as an extension of the XQuery language that connects XQuery with SPARQL and OWL reasoners. XQOWL embeds SPARQL queries (via Jena SPARQL engine) into XQuery and enables to make calls to OWL reasoners (Hermit, Pellet and FaCT++) from XQuery. It permits to combine queries against XML and RDF/OWL resources as well as to reason with RDF/OWL data. Therefore input data can be either XML or RDF/OWL and output data can be formatted in XML (also using RDF/OWL XML serialization).

Thus the framework proposes to embed SPARQL code into XQuery as well as to make calls to OWL reasoners from XQuery. With this aim a *Java API* has been implemented on top of the OWL API [11] and OWL Reasoner API [10] that makes possible to interconnect XQuery with SPARQL and OWL reasoners. The Java API is invoked from XQuery thanks to the use of the *Java Binding* facility available in most of XQuery processors (this is the case, for instance, of *BaseX* [8], *Exist* [14] and *Saxon* [13]). The Java API enables to connect XQuery to Hermit, Pellet and FaCT++ reasoners as well as to Jena SPARQL engine. The Java API returns the results of querying and reasoning in XML format which can be handled from XQuery. It means that querying and reasoning RDF/OWL with XQOWL one can give XML format to results in either XML or RDF/OWL. In particular, lifting and lowering is possible in XQOWL.

Therefore our proposal can be seen as an extension of the proposed approaches for combining SPARQL and XQuery. Our XQOWL framework is mainly focused on the use of XQuery for querying and reasoning with OWL ontologies. It makes

Table 1. Java API of XQOWL

Java API
public OWLReasoner getOWLReasonerHermiT(OWLOntology ontology)
public OWLReasoner getOWLReasonerPellet(OWLOntology ontology)
public OWLReasoner getOWLReasonerFact(OWLOntology ontology)
public String OWLSPARQL(String filei, String queryStr)
public <T extends OWLAxiom> String OWLQuerySetAxiom(Set<T> axioms)
public <T extends OWLEntity> String[] OWLQuerySetEntity(Set<T> elems)
public <T extends OWLEntity> String[] OWLReasonerNodeEntity(Node <T> elem)
public <T extends OWLEntity> String[] OWLReasonerNodeSetEntity(NodeSet<T> elems)

possible to write complex queries that combines SPARQL queries with reasoning tasks. As far as we know our proposal is the first to provide such a combination.

The implementation has been tested with the BaseX processor [8] and can be downloaded from our Web site <http://indalog.ua.es/XQOWL>. There the XQOWL API and the examples of the paper are available as well as installation instructions.

Let us remark that here we continue our previous works on combination of XQuery and the Semantic Web. In [1] we have described how to extend the syntax of XQuery in order to query RDF triples. After, in [2] we have presented a (Semantic Web) library for XQuery which makes possible to retrieve the elements of an ontology as well as to use SWRL. Here, we have followed a new direction, by embedding existent query languages (SPARQL) and reasoners into XQuery.

2 XQOWL

XQOWL allows to embed SPARQL queries into XQuery. It also makes possible to make calls to OWL reasoners. With this aim a Java API has been developed.

2.1 XQOWL: The Java API

Basically, the Java API has been developed on top of the OWL API and the OWL Reasoner API and makes possible to retrieve results from SPARQL and OWL reasoners. The elements of the library are shown in Table 1.

The first three elements of the library: *getOWLReasonerHermiT*, *getOWLReasonerPellet* and *getOWLReasonerFact* make possible to instantiate HermiT, Pellet and FaCT++ reasoners.

The fourth element of the library *OWLSPARQL* makes possible to instantiate SPARQL Jena engine. The input of this method is an ontology included in a file and a string representing the SPARQL query. The output is a file (name) including the result of the query.

The rest of elements (i.e., *OWLQuerySetAxiom*, *OWLQuerySetEntity*, *OWLReasonerNodeSetEntity* and *OWLReasonerNodeEntity*) of the Java API make possible to handle the results of calls to SPARQL and OWL reasoners.

Table 2. Social Network Ontology (in Description Logic Syntax)

<i>Ontology</i>	
event, message \sqsubseteq activity (1)	wall, album \sqsubseteq user_item (2)
\forall created_by.activity \sqsubseteq user (3)	added_by, sent_by \sqsubseteq created_by (4)
$T \sqsubseteq \leq 1.$ belongs_to (5)	\forall belongs_to.user_item \sqsubseteq user (6)
\exists friend_of.Self $\sqsubseteq \perp$ (7)	friend_of $^{-}$ \sqsubseteq friend_of (8)
\forall friend_of.user \sqsubseteq user (9)	\forall invited_to.user \sqsubseteq event (10)
\forall recommended_friend_of.user \sqsubseteq user (11)	friend_of \cdot friend_of \sqsubseteq recommended_friend_of (12)
\exists replies_to.Self $\sqsubseteq \perp$ (13)	\forall replies_to.message \sqsubseteq message (14)
$T \sqsubseteq \leq 1.$ written_in (15)	\forall written_in.message \sqsubseteq wall (16)
\forall attends_to.user \sqsubseteq event (17)	attends_to $^{-}$ \equiv confirmed_by (18)
\forall i_like_it.user \sqsubseteq activity (19)	i_like_it $^{-}$ \equiv liked_by (20)
\forall content.message \sqsubseteq String (21)	\forall date.event \sqsubseteq DateTime (22)
\forall name.event \sqsubseteq String (23)	\forall nick.user \sqsubseteq String (24)
\forall password.user \sqsubseteq String (25)	event \sqcap \exists confirmed_by.user \sqsubseteq popular (26)
activity \sqcap \exists liked_by.user \sqsubseteq popular (27)	activity $\sqsubseteq \leq 1$ created_by.user (28)
message \sqcap event $\equiv \perp$ (29)	

OWL Reasoners implement Java interfaces of the OWL API for storing OWL elements. The main Java interfaces are *OWLAxiom* and *OWLEntity*. *OWLAxiom* is a Java interface which is a super-interface of all the types of OWL axioms: *OWLSUBClassOfAxiom*, *OWLSUBDataPropertyOfAxiom*, *OWLSUBObjectPropertyOfAxiom*, etc. *OWLEntity* is a Java interface which is a super-interface of all types of OWL elements: *OWLClass*, *OWLDatatype*, *OWLDataproerty*, etc.

The XQOWL API includes the method *OWLQuerySetAxiom* that returns a file name where a set of axioms are included. It also includes *OWLQuerySetEntity* that returns in an array the URI's of a set of entities. Moreover, *OWLReasonerNodeEntity* returns in an array the URI's of a node. Finally, *OWLReasonerNodeSetEntity* returns in an array the URIs of a set of nodes.

2.2 XQOWL: SPARQL

XQOWL is an extension of the XQuery language. Firstly, XQOWL allows to write XQuery queries in which calls to SPARQL queries are achieved and the results of SPARQL queries in XML format (see [3]) can be handled by XQuery.

In XQOWL, XQuery variables can be bounded to results of SPARQL queries and vice versa, XQuery bounded variables can be used in SPARQL expressions. Therefore, in XQOWL both XQuery and SPARQL queries can share variables.

Let us suppose an ontology about a social network (see Tables 2 and 3) and the following query returns the individuals of concepts *user* and *event* in the social network:

Table 3. Individuals and object/data properties of the ontology

Ontology Instance
user(jesus), nick(jesus,jalmen), password(jesus,passjesus), friend_of(jesus,luis)
user(luis), nick(luis,lamluis), password(luis,luis0000)
user(vicente), nick(vicente,vicente), password(vicente,vicvicvic), friend_of(vicente,luis), i_like_it(vicente,message2), invited_to(vicente,event1), attends_to(vicente,event1)
event(event1), added_by(event1,luis), name(event1,"Next conference"), date(event1,21/10/2014)
event(event2)
message(message1), sent_by(message1,jesus), content(message1,"I have sent the paper")
message(message2), sent_by(message2,luis), content(message2,"good luck!"), replies_to(message2,message1)
wall(wall_jesus), belongs_to(wall_jesus,jesus)
wall(wall_luis), belongs_to(wall_luis,luis)
wall(wall_vicente), belongs_to(wall_vicente,vicente)

```

declare namespace spql="http://www.w3.org/2005/sparql-results#";
declare namespace xqo = "java:xqowl.XQOWL";

let $model := "socialnetwork.owl"
for $class in ("sn:user", "sn:event")
return
let $queryStr := concat(
    "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX sn: <http://www.semanticweb.org/socialnetwork.owl#>
    SELECT ?Ind
    WHERE { ?Ind rdf:type ", $class, " }")
return
let $xqo := xqo:new()
let $res:= xqo:OWLSPARQL($xqo,$model,$queryStr)
return
doc($res)/spql:sparql/spql:results/spql:result/spql:binding/spql:uri/text()

```

Let us observe that the name of the classes (i.e., `sn:user` and `sn:event`) is defined by an XQuery variable (i.e., `$class`) in a `for` expression, which is passed as parameter of the SPARQL expression. In addition, the result is obtained in an XQuery variable (i.e. `$res`). Here *OWLSPARQL* of the XQOWL API is used to call the SPARQL Jena engine, which returns a file name (a temporal file) in which the result is found. Now, `$res` can be used from XQuery to obtain the URIs of the elements:

```
doc($res)/spql:sparql/spql:results/spql:result/spql:binding/spql:uri/text()
```

In this case, we obtain the following plain text:

```
http://www.semanticweb.org/socialnetwork.owl#vicente
http://www.semanticweb.org/socialnetwork.owl#jesus
http://www.semanticweb.org/socialnetwork.owl#luis
http://www.semanticweb.org/socialnetwork.owl#event2
http://www.semanticweb.org/socialnetwork.owl#event1
```

2.3 XQOWL: OWL API and Reasoners

XQOWL can be also used for querying and reasoning with OWL. With this aim the OWL API and OWL Reasoner API have been integrated in XQuery. Also for this integration, the XQOWL API is required. For using OWL API and Reasoners from XQOWL there are some calls to be made from XQuery code. Firstly, we have to instantiate the ontology manager by using *createOWLontologyManager*; secondly, the ontology has to be loaded by using *loadOntologyFromOntologyDocument*; thirdly, in order to handle OWL elements we have to instantiate the data factory by using *getOWLDataFactory*; finally, in order to select a reasoner *getOWLReasonerHermiT*, *getOWLReasonerPellet* and *getOWLReasonerFact* are used.

For instance, let us suppose now we query the object properties of the ontology using the OWL API as follows:

```
let $xqo := xqo:new(),
    $man := api:createOWLontologyManager(),
    $fileName := file:new($file),
    $ont := om:loadOntologyFromOntologyDocument($man,$fileName)
return
doc(xqo:OWLQuerySetAxiom($xqo,o:getAxioms($ont)))/rdf:RDF/
owl:ObjectProperty
```

obtaining the following result:

```
<ObjectProperty...rdf:about="...#added_by">
  <rdfs:subPropertyOf rdf:resource="...#created_by"/>
  <rdfs:domain rdf:resource="...#event"/>
  <rdfs:range rdf:resource="...#user"/>
</ObjectProperty>
<ObjectProperty ... rdf:about="...#attends_to">
  <inverseOf rdf:resource="...#confirmed_by"/>
  <rdfs:range rdf:resource="...#event"/>
  <rdfs:domain rdf:resource="...#user"/>
</ObjectProperty>
...
```

Let us suppose now we want to retrieve instances of concepts **activity** and **user**. Now, we can write the following query using the HermiT reasoner:

```
for $classes in ("activity","user")
let $xqo := xqo:new(),
    $man := api:createOWLontologyManager(),
    $fileName := file:new($file),
    $ont := om:loadOntologyFromOntologyDocument($man,$fileName),
    $fact := om:getOWLDataFactory($man),
    $iri := iri:create(concat($base,$classes)),
    $reasoner := xqo:getOWLReasonerHermiT($xqo,$ont),
    $class := df:getOWLClass($fact,$iri),
    $result:= r:getInstance($reasoner,$class,false()),
```

```

    $dispose := r:dispose($reasoner)
return
<concept class="{$classes}">
{for $instances in
    xqo:OWLReasonerNodeSetEntity($xqo,$result)
    return
        <instance>{substring-after($instances, '#')}</instance>
}
</concept>

```

obtaining the following result in XML format:

```

<concept class="activity">
    <instance>message1</instance>
    <instance>message2</instance>
    <instance>event1</instance>
    <instance>event2</instance>
</concept>
<concept class="user">
    <instance>jesus</instance>
    <instance>vicente</instance>
    <instance>luis</instance>
</concept>

```

Here *getInstances* of the OWL Reasoner API is used to retrieve the instances of a given ontology class. In addition, a call to *create* of the OWL API, which creates the IRI of the class, and a call to *getClass* of the OWL API, which retrieves the class, are required. The OWL Reasoner is able to deduce that **message1** and **message2** belongs to concept **activity** since they belongs to concept **message** and **message** is a subconcept of **activity**. The same can be said for events.

3 Conclusions and Future Work

In this paper we have presented an extension of XQuery called XQOWL to query XML and RDF/OWL documents, as well as to reason with RDF/OWL resources. As future work, we would like to extend our work as follows. Firstly, we would like to extend our Java API. More concretely, with the SWRL API in order to execute rules from XQuery, and to be able to provide explanations about ontology inconsistence. Secondly, we would like to use our framework in ontology transformations (refactoring, coercion, splitting, amalgamation) and matching.

References

1. Almendros-Jiménez, J.M.: Extending XQuery for Semantic Web Reasoning. In: Abreu, S., Seipel, D. (eds.) INAP 2009. LNCS, vol. 6547, pp. 117–134. Springer, Heidelberg (2011)
2. Almendros-Jiménez, J.M.: Querying and Reasoning with RDF(S)/OWL in XQuery. In: Du, X., Fan, W., Wang, J., Peng, Z., Sharaf, M.A. (eds.) APWeb 2011. LNCS, vol. 6612, pp. 450–459. Springer, Heidelberg (2011)
3. Beckett, D., Broekstra, J.: SPARQL Query Results XML Format, 2nd edn. (2013), <http://www.w3.org/TR/rdf-sparql-XMLres/>

4. Bikakis, N., Tsinaraki, C., Stavrakantonakis, I., Gioldasis, N., Christodoulakis, S.: The SPARQL2XQuery interoperability framework. *World Wide Web*, 1–88 (2013)
5. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. *Journal on Data Semantics* 1(3), 147–185 (2012)
6. Droop, M., Flarer, M., Groppe, J., Groppe, S., Linnemann, V., Pinggera, J., Santner, F., Schier, M., Schöpf, F., Staffler, H., Zugal, S.: Bringing the XML and semantic web worlds closer: Transforming XML into RDF and embedding xPath into SPARQL. In: Filipe, J., Cordeiro, J. (eds.) *Enterprise Information Systems. LNBI*, vol. 19, pp. 31–45. Springer, Heidelberg (2009)
7. Groppe, S., Groppe, J., Linnemann, V., Kukulenz, D., Hoeller, N., Reinke, C.: Embedding SPARQL into XQUERY/XSLT. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 2271–2278 (2008)
8. Grun, C.: BaseX. The XML Database (2014), <http://basex.org>
9. Haarslev, V., Möller, R., Wandelt, S.: The revival of structural subsumption in tableau-based description logic reasoners. In: Proceedings of the 2008 International Workshop on Description Logics (DL 2008), pp. 701–706. CEUR-WS (2008)
10. Horridge, M.: OWL Reasoner API (2009), <http://owlapi.sourceforge.net/javadoc/org/semanticweb/owlapi/reasoner/OWLReasoner.html>
11. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. *Semantic Web* 2(1), 11–21 (2011)
12. Horrocks, I., Motik, B., Wang, Z.: The Hermit OWL Reasoner. In: Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012), Manchester, UK (2012)
13. Kay, M.: Ten reasons why saxon xquery is fast. *IEEE Data Eng. Bull.* 31(4), 65–74 (2008)
14. Meier, W.: eXist: An open source native XML database. In: Chaudhri, A.B., Jeckle, M., Rahm, E., Unland, R. (eds.) *NODE-WS 2002. LNCS*, vol. 2593, pp. 169–183. Springer, Heidelberg (2003)
15. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* 5(2), 51–53 (2007), <http://dx.doi.org/10.1016/j.websem.2007.03.004>
16. Staab, S., Studer, R.: *Handbook on ontologies*. Springer (2010)
17. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Furbach, U., Shankar, N. (eds.) *IJCAR 2006. LNCS (LNAI)*, vol. 4130, pp. 292–297. Springer, Heidelberg (2006)

Deriving Folksonomies for Improving Web API Search

Devis Bianchini

Dept. of Information Engineering University of Brescia
Via Branze, 38 - 25123 Brescia, Italy
`devis.bianchini@unibs.it`

Abstract. Web APIs, that is, software components made available by third parties through web interfaces, can be aggregated to develop web applications, also known as mashups. Also in this application domain, tagging performed by other mashup designers, who used available Web APIs and mashups composed of them, might be exploited as knowledge that progressively emerges from the community of designers. Web API tagging has some peculiar aspects that will be analyzed in this paper. On the other hand, folksonomies are Web 2.0 tools for conceptualizing knowledge emerging from the bottom. In this paper, we discuss the adoption of folksonomy concepts in modeling Web API use for mashup development. We motivate the adoption of folksonomies in this context and we present the differences with other models that represent very close information. Our folksonomy model is meant to be fully compliant with existing and commonly used public Web API repositories. It is not intended to substitute them, but to complement their contents in order to enable advanced Web API search facilities in such a collaborative environment.

1 Introduction

Folksonomies are a popular tool for annotating and categorizing contents with tags created and managed collaboratively. This tool requires less effort for its creation and management, and this feature determined its recent success and diffusion, in particular compared to Semantic Web ontologies, that are usually created by few experts, are more consistent and formal than folksonomies, but are also relatively static and require big maintenance efforts. The common definition of a folksonomy views it as a set of triplets, composed of *tags* assigned by *users* to *resources*, like web pages, images, videos. Social tagging systems such as Flickr¹, designed for sharing pictures, or del.icio.us², for social bookmarking, adopt these tools (and this definition) for their purposes. The use of the same (or similar) tags by more than one user to annotate the same resources can be viewed as "knowledge emerging from the bottom", in the form of a *collective classification schema* [21], assigned by individuals who act independently. A

¹ <http://www.flickr.com/>

² <http://del.icio.us/>

folksonomy for Web APIs could be described in the same manner, where *tags* are assigned by *mashup designers* to the *Web APIs*, in order to enable other designers to select them for developing their own web applications. In this sense, the kind of knowledge emerging here is the same as the one coming from tagging of a generic web resource. Moreover, other approaches proposed classical folksonomy models to share software components and services [7,24]. Nevertheless, knowledge emerging from the community of designers who share their experience in using Web APIs for mashup construction can be further enriched thanks to some peculiar aspects of Web API tagging. In fact, a Web API may be tagged differently while used in different mashups. Therefore, tags assigned by designers to Web APIs within different mashups assume a distinct relevance, that is, can be "contextualized" with respect to the mashups where they have been assigned. This aspect may be exploited during Web API search to better refine search results. These considerations bring to a Web API folksonomy model that differs from the traditional representation of folksonomies as triplets.

In this paper we propose an innovative Web API folksonomy model and we show how to use it for supporting Web API search and aggregation. Specifically, given a set of Web APIs whose descriptions are extracted from a public repository, where each Web API is described through different kinds of tags assigned by mashup designers and is associated to the mashups where the API has been included and tagged, we will describe how we can model all this information as a folksonomy, in order to identify proper relationships between tags and exploit both the folksonomy structure and tag relationships for improving Web API search and aggregation. A tool for explorative search over the repository of Web APIs, based on the folksonomy, is presented as well.

The paper is organized as follows. Section 2 contains the definitions of the Web API folksonomy model. Exploitation of this model for supporting Web API search and aggregation is described in Sections 3 and 4, respectively. In Section 5 we present the tool for explorative search over a well known repository of Web APIs. Experimental evidences are described in Section 6, while in Section 7 a comparison with related work in literature is performed to highlight cutting-edge features of our proposal. Finally, Section 8 closes the paper.

2 The Folksonomy Model

A folksonomy is used to model a classification "emerging from the bottom", performed by users who collaboratively tag and annotate shared resources [15]. According to this definition, folksonomies are social tagging tools, where *mashup designers*, $\mathcal{D} = \{d_1, d_2, \dots, d_k\}$, assign *tags*, $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$, to *resources*, that is, *Web APIs*, $\mathcal{W} = \{w_1, w_2, \dots, w_n\}$. In these tagging systems, designers annotate Web APIs with tags, hence creating ternary associations between the set of designers, the set of tags and the set of Web APIs. Therefore, a folksonomy can be defined by a set of annotations $\mathcal{A} \subseteq \mathcal{D} \times \mathcal{T} \times \mathcal{W}$. Each annotation can be considered to be a hyperedge in a graph, where the nodes are the mashup designers, the tags and the resources, and where each hyperedge represents the

fact that a given designer associated a certain API with a certain tag. According to this viewpoint, the folksonomy can be formally defined as an hypergraph as follows.

Definition 1. *A folksonomy is represented as an hypergraph, $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \mathcal{D} \cup \mathcal{W}$ is the set of vertices, and $\mathcal{E} = \{(d, t, w) | (d, t, w) \in \mathcal{A} \subseteq \mathcal{D} \times \mathcal{T} \times \mathcal{W}\}$ is the set of hyperedges of the hypergraph.*

Definition 1 corresponds to the classical definition of folksonomy. Nevertheless, in the context of Web API discovery and aggregation into mashups, additional aspects should be considered:

- within public repositories, Web APIs are further specified through other properties, such as the categories and a set of technical features (like protocols, data formats, SSL support and authentication mechanisms); these features could be considered as special kinds of tags, taken from pre-defined tag sets (the taxonomy of the categories, that represents a top-down classification of resources and is supplied by the owner of the repository where Web APIs have been registered, or the pre-defined set of technical feature values, such as XML or JSON for the data format, distinguished with respect to the kind of feature, such as protocols, data formats, SSL support and authentication mechanisms); these specific kinds of tags are assigned by special users, i.e., the providers of the Web APIs, who not necessarily are mashup designers, but, as third parties who make available Web APIs as software components over the network, own a deeper knowledge on how the APIs work; hereafter in this paper, as technical features we consider protocols, data formats, SSL support and authentication mechanisms, since they have been used to characterize Web APIs in a well known public repository, **ProgrammableWeb**³, that is the one we considered in our experimental evaluation (see Section 6); however, the model can be easily extended to other kinds of technical features as well;
- usually, the set of tags \mathcal{T} in a folksonomy are unrelated and \mathcal{T} is completely unstructured; however, this brings to problems like *tag ambiguity*, *discrepancy in granularity* and *lack of synonymy*; these limitations do not apply to tags representing categories and technical feature values, that are taken from pre-defined tag sets, whose meaning could be considered as shared within the community of mashup designers; *semantic tagging*, that is, tagging refined through a sense disambiguation process, has been proposed to overcome these limitations for the remaining tags, other than categories and technical feature values; there are many solutions for tag disambiguation, either explicit [4] or implicit [21], that could be applied to integrate a few semantics within folksonomies; note that the application of tag disambiguation techniques is different and less expensive than managing for ensuring a fully consistent ontology over the whole resource space; designers may use *faceted ontologies*, that is, partial or small ontologies conceptualizing specific facets of knowledge in the form of semantic tags, but this is limited to

³ <http://www.programmableweb.com>

the tags they used and the approach does not impose the use of a single, monolithic ontology [21]; possible similarities between tags across different faceted ontologies might be inferred using ontology alignment or matching techniques [22];

- based on a semantic disambiguation of tags, either implicit or explicit as underlined above, tag hierarchies can be introduced to enhance the expressivity of the folksonomy model, and also to keep succinct the tag-based descriptions; a tag hierarchy \mathcal{H} can be defined as a directed acyclic graph (DAG) where vertices are tags and edges represent *sub-tag relationships*; if an explicit semantic disambiguation of tags has been performed, identification of sub-tag relationships may be obtained by applying well known techniques, such as the ones based on WordNet, techniques deriving from reasoning based on the semantic characterization of tags [22], or techniques such as the ones proposed in [21]; otherwise, different techniques based on associative semantics [7] are required;
- contextualization of the tagging procedure with respect to the mashup where it is performed may be another relevant innovation introduced for a folksonomy model in the domain of Web API selection and aggregation; according to this perspective, a tag may be assigned to a Web API with respect to a given mashup; for instance, Web APIs such as Google Maps are often used in mashups referring to different domains and, therefore, tagged in different manner.

Therefore, compared to Definition 1, we revise here the model of a folksonomy of Web APIs. According to this definition, folksonomies are social tagging tools, where *mashup designers*, $\mathcal{D} = \{d_1, d_2, \dots, d_k\}$, assign *tags* to *resources*, that is, *Web APIs*, $\mathcal{W} = \{w_1, w_2, \dots, w_n\}$. Tags are chosen from a set $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$. Tags may be *semantic tags*, that is, defined with respect to a classification/thesaurus/ontology used to qualify/disambiguate the tags belonging to the set \mathcal{T} . Let be $\mathcal{S} = \{s_1, s_2, \dots, s_h\}$ a finite set of such classifications/thesauri/ontologies. Finally, tags can be assigned with respect to a mashup, taken from a finite set $\mathcal{M} = \{m_1, m_2, \dots, m_j\}$. In this revised tagging system, designers annotate Web APIs with (semantic) tags within a given mashup, hence creating 5-ary associations between the set of designers, the set of tags, the set of classifications/thesauri/ontologies, the set of Web APIs and the set of mashups. Therefore, a Web API folksonomy can be defined by a set of annotations $\mathcal{A}^{ext} \subseteq \mathcal{D} \times \mathcal{T} \times \mathcal{S} \times \mathcal{W} \times \mathcal{M}$. Each annotation can be considered to be a hyperedge in a graph, where the nodes are the mashup designers, the tags, references to a classification/thesaurus/ontology, the Web APIs and the mashups, and where each hyperedge represents the fact that a given designer associated a certain API with a certain (semantic) tag with respect to a given mashup. Therefore, the Web API folksonomy can be formally defined as follows.

Definition 2. A Web API folksonomy is defined as an hypergraph, $\mathcal{G}^{ext} = \langle \mathcal{V}^{ext}, \mathcal{E}^{ext} \rangle$, where $\mathcal{V}^{ext} = \mathcal{D} \cup \mathcal{T} \cup \mathcal{S} \cup \mathcal{W} \cup \mathcal{M}$ is the set of vertices, $\mathcal{E}^{ext} = \{(d, t, s, w, m) | (d, t, s, w, m) \in \mathcal{A}^{ext} \subseteq \mathcal{D} \times \mathcal{T} \times \mathcal{S} \times \mathcal{W} \times \mathcal{M}\}$ is the set of hyperedges of the hypergraph.

Example. In Figure 1 we reported an instance of the Web API folksonomy, where the Flickr API has been tagged within the Imaginalaxy mashup by a designer $d_1 \in \mathcal{D}$ using the **sharing** and **picture** tags. In this example, we considered semantic characterization of tags based on WordNet terminological system, whose intended semantics is given in Table 1. In WordNet, terms are associated to sets of synonyms (*synsets*). Each synset has a unique code to unambiguously identify the definition of the term. Codes have a complex structure and for clarity reasons they have been substituted with **synsetcode:x** in Figure 1 and in Table 1. In a synset, each tag $t \in \mathcal{T}$, whose semantics is disambiguated through WordNet, is described by: (i) the name n_t of the tag itself; (ii) all the synonyms of t ; (iii) the human readable definition d_t associated with the synset. As explained in Section 5, we provide the designers with a wizard that supports them in semantic disambiguation: the wizard (both during annotation and search) displays the human-readable definition of the term in WordNet and, after the designers selected the intended meaning, the synset code is automatically assigned to the term to identify its meaning. We remark here that designers act independently each other during the assignment of tags to Web APIs, according to their own knowledge on mashup development only. In this particular example, we assumed that designers agreed upon the use of WordNet for tag disambiguation, but this is just a case study and it is not mandatory in our approach. In this example, categories and technical features, as special kinds of tags assigned by Web API providers, have not been shown, but they can be represented in a similar way. For example, categories are represented like **sharing** or **snapshot** tags in Figure 1 and a reference to the classification where categories have been taken from (e.g., ProgrammableWeb pre-defined set of categories) is stored as well.

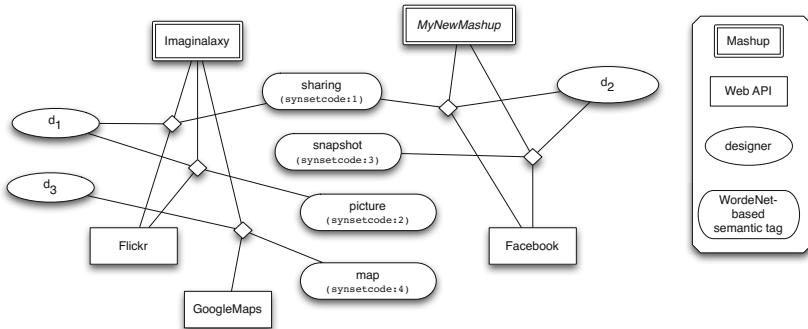


Fig. 1. An instance of the Web API folksonomy

2.1 Tag Hierarchies

The definition of a hierarchy between tags, by establishing *sub-tag relationships* between them, will be of extreme relevance for enhancing Web API search and

Table 1. The definition of semantic tags used in the running example based on the WordNet lexical system

Synset code	Tag name	Synonyms	Definition
synsetcode:1	sharing	{}	using or enjoying something jointly with others
synsetcode:2	picture	{photograph, exposure, photo, pic}	a representation of a person or scene in the form of a print or transparent slide or in digital format
synsetcode:3	snapshot	{snap, shot}	an informal photograph; usually made with a small hand-held camera
synsetcode:4	map	{}	a diagrammatic representation of the earth's surface (or part of it)

aggregation, as explained in the next sections. If tags have been associated to an explicit semantics, through a classification/vocabulary/ontology \mathcal{S} , then traditional subsumption checking techniques can be applied to determine if there exists a sub-tag relationship between tags [4,22]. For instance, if a tag is disambiguated according to the WordNet lexical system, *hypernymy* and *hyponymy* terminological relationships defined in WordNet can be exploited to determine the sub-tag relationships, as extensively described in [4]. Otherwise, if no semantics is defined on tags, then an associative semantics perspective is assumed to establish if there exists a sub-tag relationship. According to this perspective, if x is a sub-tag of y , then x is commonly associated with y by the community of users and, if a Web API is tagged with x , then it may also be tagged with y .

Definition 3. Given two tags $t_1, t_2 \in \mathcal{T}$, t_1 is a sub-tag of t_2 , denoted with $t_1 \prec t_2$, if one of the following conditions holds:

- t_1 and t_2 are terms disambiguated through WordNet and there exists a *hypernymy/hyponymy* relationship between them, as described in [4];
- t_1 and t_2 are associated to ontological concepts in the same ontology $s_i \in \mathcal{S}$ and a subclass relationship can be inferred between them using one of the available subsumption checking techniques [22];
- t_1 and t_2 are values of a technical feature (namely, protocol, data format, authentication mechanism, SSL support), or they both belong to the same categorization and all the Web APIs annotated with t_1 are also annotated with t_2 , i.e., $(d, t_1, s, w, m) \in \mathcal{A}^{ext} \wedge t_1 \prec t_2 \Rightarrow (d, t_2, s, w, m) \in \mathcal{A}^{ext}$;
- in all the other cases, if all the Web APIs annotated with t_1 are also annotated with t_2 , i.e., $(d, t_1, s, w, m) \in \mathcal{A}^{ext} \wedge t_1 \prec t_2 \Rightarrow (d, t_2, s, w, m) \in \mathcal{A}^{ext}$.

We assume, according to Definition 3, that the sub-tag relationship is transitive, that is, if $t_1 \prec t_2$ and $t_2 \prec t_3$, then $t_1 \prec t_3$, $\forall t_1, t_2, t_3 \in \mathcal{T}$, and reflexive, that is, $t \prec t$, $\forall t \in \mathcal{T}$.

Example. In WordNet, synsets are related through *hypernymy* and *hyponymy* terminological relationships, used to represent generalization/specialization relationships between terms. In Figure 2 a portion of WordNet vocabulary is shown.

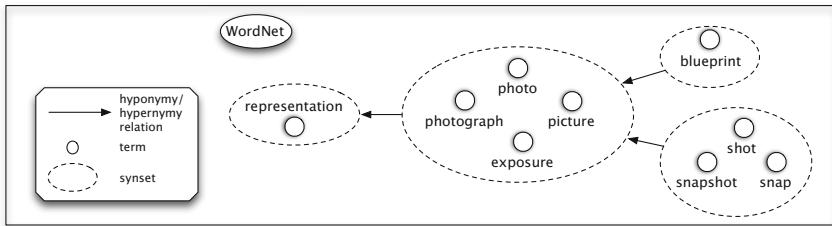


Fig. 2. An example of *hypernymy* and *hyponymy* terminological relationships between synsets in WordNet

In this example, the following sub-tag relationships hold: `snapshot` \prec `photo`, `snapshot` \prec `picture`, `snapshot` \prec `representation` (by transitivity). Let's suppose now that `forecasting` and `weather` tags have not been semantically disambiguated, therefore `forecasting` \prec `weather` implies, according to associative semantics, that in the Web API repository any API tagged with `forecasting` is also tagged with `weather`.

2.2 Data Storage of Folksonomy Contents and Tag Hierarchies

To fasten basic operations on folksonomy contents and tag hierarchies, proper data structures have been chosen to storage the 5-uples of the folksonomy and sub-tag relationships. This will bring advantages in terms of response time during Web API search, as experimentally verified (see Section 6). A formal analysis of the computational properties of the approach presented in this paper is part of future extensions of the work. In particular, all the annotations in \mathcal{A}^{ext} are stored in a table in order to enable basic operations featured for relational databases to retrieve folksonomy contents. For example, Table 2 lists the annotations of the folksonomy shown in Figure 1. Specifically, we define the following basic operations:

- the *selection of an annotation* $a \in \mathcal{A}^{ext}$, that fits a given condition $cond$, denoted with $\sigma_{cond}(\mathcal{A}^{ext})$; if the condition $cond$ is simply expressed as $x \in X$ (or using the shortcut x), where $X = \mathcal{D}|\mathcal{T}|\mathcal{S}|\mathcal{W}|\mathcal{M}$, this means selecting an annotation containing the element $x \in X$; for instance, $\sigma_w(\mathcal{A}^{ext})$, where $w \in \mathcal{W}$, selects all the annotations in \mathcal{A}^{ext} that involve the Web API $w \in \mathcal{W}$ ($\sigma_{\text{Flickr}}(\mathcal{A}^{ext})$ returns all annotations referring to the Flickr API);
- the *projection over the columns of the annotation table*, denoted with $\pi_X(\mathcal{A}^{ext})$, to consider, for all the annotations, only the values on the columns corresponding to the sets in X ; for instance, $\pi_W(\mathcal{A}^{ext})$ lists all the annotated Web APIs (for the example shown in Figure 1 and Table 2, this operation returns {Flickr, Facebook, Google Maps}).

Basic operations can be composed in different ways, as will be described in the following sections.

Table 2. Tabular representation of annotations for the sample folksonomy used in the running example

\mathcal{D}	\mathcal{T}	\mathcal{S}	\mathcal{W}	\mathcal{M}
d_1	sharing - synsetcode:1	WordNet	Flickr	Imaginalaxy
d_1	picture - synsetcode:2	WordNet	Flickr	Imaginalaxy
d_2	sharing - synsetcode:1	WordNet	Facebook	MyNewMashup
d_2	snapshot - synsetcode:3	WordNet	Facebook	MyNewMashup
d_3	map - synsetcode:4	WordNet	Google Maps	Imaginalaxy

Sub-tag relationships have been represented through an hash table, where each tag is assigned to a key composed as `prefix+tagCode`. The `prefix` part is an integer corresponding to the classification/thesaurus/ontology $s_i \in \mathcal{S}$ used to semantically characterize the tag, among the following options: top-down categorization, WordNet, ontology (to each specific ontology a different `prefix` is assigned), protocol feature, data format feature, authentication mechanism feature, SSL support feature, none. Within the same prefix, the `tagCode` integer reflects the sub-tag relationships between tags, that is, $\text{tagCode}(t_1) \leq \text{tagCode}(t_2) \Rightarrow t_1 \prec t_2$. Therefore, sub-tag relationship checking is performed as an integer comparison, while sub-tag relationships can be easily updated off-line, after new tags are added by designers to Web APIs.

3 Folksonomy-Based Web API Search

The folksonomy model described above can be exploited for Web API search. Designers can issue requests for Web API search by using tags to specify the desired categories and technical features. Furthermore, the designers who are looking for a Web API may also specify the mashup where the API is being included. This mashup can be specified through the set of Web APIs that have been already included into it. The presence of the mashup under development within the request is of paramount relevance, since enables the discovery algorithm to rely on past occurrences of the same APIs within existing mashups to properly rank search candidates. For example, if we consider the `ProgrammableWeb` repository, two Web APIs such as `Twitter` and `GoogleMaps` have been used together in about 500 mashups, while `LinkedIn` and `GoogleMaps` APIs have been used in no more than 30 mashups⁴. This would suggest the use of `Twitter` API for posting and displaying comments from social networks, instead of `LinkedIn`, in a mashup where `GoogleMaps` has been already inserted. Formally, a Web API request w^r is defined as follows.

Definition 4. A Web API request is formally represented as $w^r = \langle \mathcal{C}^r, \mathcal{T}^r, \mathcal{F}^r, m^r \rangle$, where: (i) \mathcal{C}^r is the set of required categories; (ii) \mathcal{T}^r is the set of required tags (other than categories and technical feature values); (iii) \mathcal{F}^r are

⁴ Last update on July 31st, 2014.

the required technical features, to be distinguished among protocols, data formats, SSL support and authentication mechanism; for each kind of feature, a distinct vector is inserted in the request w^r ; (iv) m^r is the mashup where the Web API to search for will be inserted, in turn characterized through the set of Web APIs $\{w_m^r\}$ already included in the mashup m^r . All the elements are optional in the request. The presence of the m^r specification in the request w^r depends on the search target. If the goal is to select a single Web API (for instance, to start a new mashup), then no APIs $\{w_m^r\}$ have been previously selected yet and m^r is not specified. Otherwise, if the target is the completion of an existing mashup or the substitution of a Web API in a given mashup, then m^r represents the mashup where the Web API that is being searched will be inserted.

Example. The request $w_1^r = \langle -, \mathcal{T}^r, -, \{\text{GoogleMaps}\} \rangle$ is formulated to look for a Web API to be inserted into a mashup that already contains the Google Maps API. Tags in \mathcal{T}^r are used to search for the API. In this specific example, no requirements are made on the set of categories or on technical features.

In order to define query satisfaction, we first introduce the notion of a subsumption relationship between two sets of tags, taking into account that in our folksonomy model tags are further distinguished by their type, namely, categories, technical features (among protocols, data formats, SSL support, authentication mechanism), (semantic) tags. Given the definition of subsumption between tag sets, we will be ready to provide the notion of *Web API search outcome*.

Definition 5. Let X and Y two sets of tags, i.e., $X, Y \subseteq \mathcal{T} \cup \mathcal{C} \cup \mathcal{F}$. We state that X is subsumed by Y (that is, X is more specific than Y), denoted with $X \sqsubseteq Y$, iff, for each tag $y \in Y$, there exists a tag $x \in X$ that corresponds to y or is a sub-tag of y , that is, $x = y$ or $x \prec y$.

Example. Consider the portion of WordNet vocabulary shown in Figure 2 and the corresponding sub-tag relationships that it implies. Therefore, $\{\text{snapshot}, \text{blueprint}\} \sqsubseteq \{\text{photo}, \text{representation}\}$ follows.

Definition 6. Let $w^r = \langle \mathcal{C}^r, \mathcal{T}^r, \mathcal{F}^r, m^r \rangle$ a Web API request, let $\mathcal{C}^w \subseteq \mathcal{T}$ the set of categories associated to the Web API $w \in \mathcal{W}$, let $\mathcal{T}^w \subseteq \mathcal{T}$ the set of (semantic) tags associated to the Web API $w \in \mathcal{W}$, let $\mathcal{F}_P^w \subseteq \mathcal{T}$ the set of protocols associated to the Web API $w \in \mathcal{W}$, let $\mathcal{F}_{DF}^w \subseteq \mathcal{T}$ the set of data formats associated to the Web API $w \in \mathcal{W}$, let $\mathcal{F}_{AM}^w \subseteq \mathcal{T}$ the set of authentication mechanisms associated to the Web API $w \in \mathcal{W}$, let $\mathcal{F}_{SSL}^w \in \{\text{true}, \text{false}\}$ the SSL support associated to the Web API $w \in \mathcal{W}$. The set of search results for the request w^r is defined as $\pi_{\mathcal{W}}[\sigma_{cond^r}(\mathcal{A}^{ext})] = \{w \in \mathcal{W} | cond^r = \text{true}\}$. The condition $cond^r$ holds iff all the following statements are verified:

- the set of categories assigned to $w \in \mathcal{W}$ corresponds or is subsumed by the set of categories specified in the request, that is, $\mathcal{C}^w \sqsubseteq \mathcal{C}^r$;
- the set of (semantic) tags assigned to $w \in \mathcal{W}$ corresponds or is subsumed by the set of (semantic) tags specified in the request, that is, $\mathcal{T}^w \sqsubseteq \mathcal{T}^r$;

- the set of protocols assigned to $w \in \mathcal{W}$ corresponds or is subsumed by the set of protocols specified in the request, that is, $\mathcal{F}_P^w \sqsubseteq \mathcal{F}_P^r$;
- the set of data formats assigned to $w \in \mathcal{W}$ corresponds or is subsumed by the set of data formats specified in the request, that is, $\mathcal{F}_{DF}^w \sqsubseteq \mathcal{F}_{DF}^r$;
- the set of authentication mechanisms assigned to $w \in \mathcal{W}$ corresponds or is subsumed by the set of authentication mechanisms specified in the request, that is, $\mathcal{F}_{AM}^w \sqsubseteq \mathcal{F}_{AM}^r$;
- the SSL support specified for $w \in \mathcal{W}$ is coherent with the SSL support specified in the request, that is, $\mathcal{F}_{SSL}^w \underline{\vee} \mathcal{F}_{SSL}^r = \text{true}$, where $\underline{\vee}$ represents the binary logic operator XNOR, that returns true either if both of the members are true or if both of the members are false.

Example. Consider the sample folksonomy shown in Figure 1 and the request w_1^r , the Web APIs that fit w_1^r are the following: {Flickr, Facebook}.

Web APIs in the search results are ranked by considering the mashup m^r under construction. A Web API $w' \in \mathcal{W}$, included among the search results, is ranked better than another Web API $w'' \in \mathcal{W}$, also included among the search results, if, on average, the number of times in which w' has been included in other mashups together with the APIs already included in m^r (*average Web API co-occurrence*) is greater than the number of times in which w'' has been included in other mashups together with the APIs already included in m^r .

Definition 7. Given two Web APIs $w', w'' \in \mathcal{W}$, included among the search results, the API w' is ranked better than w'' , denoted with $w' \triangleright w''$, iff:

$$\text{avg}_{w^h \in \{w_m^r\}} \left[|\sigma_{w'}(\mathcal{A}^{ext}) \bowtie_m \sigma_{w^h}(\mathcal{A}^{ext})| \right] > \text{avg}_{w^h \in m^r} \left[|\sigma_{w''}(\mathcal{A}^{ext}) \bowtie_m \sigma_{w^h}(\mathcal{A}^{ext})| \right] \quad (1)$$

where $w^h \in \{w_m^r\}$ is one of the APIs already included in the mashup m^r under construction, $|\sigma_{w'}(\mathcal{A}^{ext}) \bowtie_m \sigma_{w^h}(\mathcal{A}^{ext})|$ denotes the number of mashups where w' and w^h have been used together ($m \in \mathcal{M}$), $|\sigma_{w''}(\mathcal{A}^{ext}) \bowtie_m \sigma_{w^h}(\mathcal{A}^{ext})|$ denotes the number of mashups where w'' and w^h have been used together ($m \in \mathcal{M}$), $\text{avg}[\cdot]$ computes the average value.

Example. Among the set of APIs {Flickr, Facebook} that fit the request w_1^r , Flickr \triangleright Facebook.

4 Folksonomy-Based Web API Aggregation

The folksonomy model of Web APIs is useful also in suggesting aggregation of Web APIs into a mashup under construction. Web API aggregation may depend on several factors, that we summarize in the concepts of *proximity* and *prominence*. The *proximity* of a Web API w with respect to another Web API w' is a measure of the degree of compatibility between the two Web APIs for being included in the same mashup, that is, a measure of how much the two

Web APIs are likely to be used together. It is based either on the number of times the two Web APIs have been used together in the same mashup or, if two Web APIs have not been included together in any mashup yet, on the *technical compatibility* of the two Web APIs. The technical compatibility is checked on the basis of the values of technical features in the two Web APIs. Feature values are compared only within the context of the same feature.

Definition 8. Let $\mathcal{F}_P^w \subseteq \mathcal{T}$ (resp., $\mathcal{F}_P^{w'} \subseteq \mathcal{T}$) the set of protocols associated to the Web API $w \in \mathcal{W}$ (resp., $w' \in \mathcal{W}$), let $\mathcal{F}_{DF}^w \subseteq \mathcal{T}$ (resp., $\mathcal{F}_{DF}^{w'} \subseteq \mathcal{T}$) the set of data formats associated to the Web API $w \in \mathcal{W}$ (resp., $w' \in \mathcal{W}$), let $\mathcal{F}_{AM}^w \subseteq \mathcal{T}$ (resp., $\mathcal{F}_{AM}^{w'} \subseteq \mathcal{T}$) the set of authentication mechanisms associated to the Web API $w \in \mathcal{W}$ (resp., $w' \in \mathcal{W}$), let $\mathcal{F}_{SSL}^w \in \{\text{true}, \text{false}\}$ (resp., $\mathcal{F}_{SSL}^{w'} \in \{\text{true}, \text{false}\}$) the SSL support associated to the Web API $w \in \mathcal{W}$ (resp., $w' \in \mathcal{W}$). The technical compatibility between two Web APIs $w \in \mathcal{W}$ and $w' \in \mathcal{W}$ is computed as follows:

$$\text{techComp}(w, w') = \frac{1}{4} \left[\sum_j \frac{2 \cdot |\mathcal{F}_j^w \cap \mathcal{F}_j^{w'}|}{|\mathcal{F}_j^w| + |\mathcal{F}_j^{w'}|} + \ell(\mathcal{F}_{SSL}^w \vee \mathcal{F}_{SSL}^{w'}) \right] \in [0, 1] \quad (2)$$

where $j = P|DF|AM$ and $\ell(\cdot) \mapsto \{0|1\}$ is a function to check the compatibility in terms of SSL support, that is, $\ell(\text{true}) = 1$ and $\ell(\text{false}) = 0$; \vee represents the binary logic operator XNOR, that returns **true** either if both of the members are **true** or if both of the members are **false**; finally, the fraction $\frac{1}{4}$ is used to normalize the $\text{techComp}(\cdot)$ value within the $[0, 1]$ range, since each of the four components of this formula have values within $[0, 1]$ range. The current formula for the technical compatibility equally weights all the technical features. Further experiments are being performed to check if the same features should be considered as more relevant, thus providing a proper setup of weighting factors to compute Equation 2.

Example. For example, if w presents **{XML, JSON, JSONP}** as data formats and **{REST}** as protocol, while w' presents **{XML, JSON}** as data formats and **{REST, Javascript, XML}** as protocols, the $\text{TechCom}()$ value is computed as:

$$\frac{1}{2} \left[\frac{2 \cdot |\{\text{XML, JSON, JSONP}\} \cap \{\text{XML, JSON}\}|}{|\{\text{XML, JSON, JSONP}\}| + |\{\text{XML, JSON}\}|} + \frac{2 \cdot |\{\text{REST}\} \cap \{\text{REST, Javascript, XML}\}|}{|\{\text{REST}\}| + |\{\text{REST, Javascript, XML}\}|} \right] \quad (3)$$

In this example, **XML** is used both as data format and as **XML-RPC** protocol and it is considered separately in the two cases. Formally, *Web API proximity* is defined as follows.

Definition 9. The proximity of a Web API $w \in \mathcal{W}$ with respect to another Web API $w' \in \mathcal{W}$, denoted with $\varrho(w, w')$, is computed as: (i) the number of mashups where w and w' have been used together ($\varrho(w, w') \geq 1$); (ii) otherwise, the degree of technical compatibility between w and w' ($\varrho(w, w') \in [0, 1]$), that is

$$\varrho(w, w') = \begin{cases} |\sigma_w(\mathcal{A}^{ext}) \bowtie_m \sigma_{w'}(\mathcal{A}^{ext})| & \text{if } |\sigma_w(\mathcal{A}^{ext}) \bowtie_m \sigma_{w'}(\mathcal{A}^{ext})| > 0 \\ \text{techComp}(w, w') & \text{otherwise} \end{cases} \quad (4)$$

where $|\sigma_w(\mathcal{A}^{ext}) \bowtie_m \sigma_{w'}(\mathcal{A}^{ext})|$ denotes the number of mashups where w and w' have been used together ($m \in \mathcal{M}$).

The notion of *prominence* is based on the definition of *proximity*. In particular, we define *prominence* of a Web API $w \in \mathcal{W}$ as the number of other Web APIs with which w has a proximity value equal or greater than a threshold th .

Definition 10. *The prominence of a Web API $w \in \mathcal{W}$, denoted with $\psi(w)$, is computed as:*

$$\psi(w) = |\{w' \in \mathcal{W} | \varrho(w, w') \geq th\}| \quad (5)$$

To establish the range for the threshold, consider that the following situations may hold between two Web APIs w and $w' \in \mathcal{W}$: (a) there exists at least one mashup where w and w' have been used together; (b) a mashup where w and w' have been used together does not exist yet, but the degree of technical compatibility between w and w' is high. In the first case, the Web API w' should be considered in the computation of the prominence for w , in the second case, since $\varrho(w, w') \in [0, 1]$, a threshold $\in [0, 1]$ is required to establish when the degree of technical compatibility between w and w' can be considered as "high". In the experimental evaluation (see Section 6) we will show how we determined the value for this threshold.

5 A Folksonomy-Driven Exploratory Tool for Web APIs

This section describes the features of **Weagle**, a tool designed for performing Web API exploration based on the folksonomy model presented in this paper. Figure 3 presents a screenshot that shows the interface of the tool. The figure shows the search and exploration mechanism, where the mashup designer may specify a set of tags for performing basic Web API search **(a)** and, optionally, may select a set of Web APIs that have been already inserted in the mashup under construction to perform advanced search **(b)**. The advanced search button opens a window, that has been implemented to guide the designer in the specification of all the elements of the request (see Definition 4) through a wizard composed of a sequence of steps (specification of the category and of the technical features, semantic disambiguation of tags, specification of Web APIs that have been already included in the mashup under construction, to enable Web API co-occurrence analysis). In particular, to guide WordNet-based semantic disambiguation, starting from the tag specified by the designer, the system queries WordNet, retrieves all the synsets that contain that term and asks the designer to select the intended meaning. Basic search corresponds to the specification of a request w^r where only \mathcal{T}^r is specified (see Definition 4). Search results, obtained as discussed in Section 3, are listed on the right **(c)**, sorted with respect to their prominence. By clicking on one of the search results, the exploration of the Web APIs according to their prominence and proximity may start within the panel on the left **(d)**. In the middle of the panel, the selected Web API is shown and, around it with a different filling color, all the other Web APIs that present a proximity value that is greater than the threshold th are rendered. The size of the Web APIs around the selected one is proportional to their degree of proximity. Before the designer performs a choice among the

Web APIs within the search results, the system by default starts displaying in the middle the Web API with the highest prominence. During exploration, the designer may click on one of the APIs in the graph and the interface refactor the graph, moving the selected API in the middle of the panel and displaying the other related APIs around it. In Figure 3 a graph with depth equal to 2 is shown. The graph depth is parametric and can be chosen by the designer before starting using the explorer.



Fig. 3. A screenshot of the Weagle explorer, used to browse Web APIs according to their degree of aggregation, based on the folksonomy model presented in this paper

6 Experimental Evaluation

We performed a preliminary evaluation based on the contents of the **ProgrammableWeb** public repository. The **ProgrammableWeb** repository is the most used and updated one for sharing Web APIs and mashups. It contains over **11,500** Web APIs, where about **1,200** of them have been added in the last year. APIs have been used in more than **7,400** mashups, while more than **2,800** mashup owners are registered in the repository. Web APIs are described through the technical features considered in this paper and are classified by means of a pre-defined categorization and designers-assigned tags. For the considered dataset, we empirically defined the threshold th for proximity identification to **0.5**. This means that, to be related through a "proximity" relationship, two Web APIs must have been used together in at least one mashup or, otherwise, they must present a technical compatibility that has to reach **0.5** or higher score, according to Equation (4). The threshold th has been set as follows. We considered different pairs $\langle w_1, w_2 \rangle$ of Web APIs registered within the **ProgrammableWeb** repository and that have been used together in at least one mashup. We evaluated the value

of technical compatibility for each pair and we chose a threshold th such that, in **80%** of cases, $TechCom(w_1, w_2) > th$.

We then performed experiments using classical IR metrics of *precision* and *recall* to evaluate experimental results [13]. The Web API search based on the folksonomy model should present a high precision (that is, the number of relevant Web APIs over the total number of APIs retrieved), but also high recall (that is, the number of relevant Web APIs retrieved over the total number of relevant APIs in the repository). In particular, high recall is required by the Weagle explorer: in fact, explorative search tools are mainly recall oriented, since their aim is at presenting as much search results as possible to enable the designer to browse them and increment his/her knowledge within the search space. To assess an high value for both precision and recall, the search process can be evaluated through the *f-measure*, that is the harmonic mean of precision and recall. The closer the *f-measure* to **1.0**, the better the search algorithm. If search has high precision, but low recall, or good recall, but low precision, then the f-measure is close to **0.0**.

We ran two kinds of experiments on an Intel laptop, with 2.53 GHz Core 2 CPU, 2GB RAM and Linux OS. Experiments have been performed ten times using different requests. To obtain a precise measure of the precision and recall, since for the experiments we need to know exactly which are the relevant Web APIs in the repository for each request, we randomly chose real mashups from the repository and we considered as relevant the Web APIs included in the mashups. We then issued the requests using the features of the Web APIs in the selected mashups and we calculated the average precision and recall of search results given by our system. In this way, we are not constrained to inspect the entire repository. The aim of these preliminary experiments is at confirming the advantages for Web API search brought by the exploitation of semantic tagging and contextualization of tag assignment with respect to the mashup where APIs have been included [6], also using a folksonomy-like model. On the other hand, providing a formalization of a Web API folksonomy may allow the application of optimized metrics such as the ones used in common folksonomies to infer the similarity between items according to tag similarity [17]. For these reasons, we compared our approach against the Web API search facilities made available within the **ProgrammableWeb** repository (that does not exploit the information of co-occurrence of Web APIs within existing mashups to enhance the search results) and against two partial implementations of our approach, where: (a) no tag hierarchies are integrated in the searching process, while contextualization of the tagging procedure with respect to the mashup where it is performed is exploited (*Weagle 1.0* prototype); (b) no contextualization of the tagging procedure with respect to the mashup, where it is performed, is considered, while tag hierarchies have been introduced (*Weagle 1.1* prototype). Comparison with other advanced approaches for Web API discovery is being performed and experimental results will be presented as future work. Table 3 shows the precision, recall and f-measure results, the standard deviation and the variance of f-measure for the compared systems. As expected, the system that presents the best f-measure value is the

complete implementation of the Weagle system (*Weagle 1.2*). Although both the exploitation of tag hierarchies and the contextualization of the tagging procedure with respect to the mashup where it is performed bring significant improvements compared to the basic searching facilities of the **ProgrammableWeb** repository, it is quite evident as the highest enhancement in f-measure value is due to the integration of tag hierarchies in the searching process. In fact, the delta in f-measure values from the **ProgrammableWeb** approach to the *Weagle 1.0* prototype is less relevant compared to the delta from the **ProgrammableWeb** approach to the *Weagle 1.0* prototype. On the other hand, tag hierarchies impact the most on the increment in recall values, while contextualization of the tagging procedure with respect to the mashup impacts the most on the precision of the search process. For what concerns response time of the search process, considering the whole Web API repository (over **11,500** APIs and more than **7,400** mashups) and a request w^r containing all the elements listed in Definition 4, the search results are returned in no more than **3 seconds** on average, that is acceptable for the considered domain of interest.

Table 3. F-measure values for the experimental evaluation

Approach	F-measure	Precision	Recall	Standard deviation	Variance
ProgrammableWeb	0,0414	0,03105	0,0621	0,0363	0,0013
Weagle v1.0	0,4247	0,5880	0,3324	0,3562	0,1269
Weagle v1.1	0,5276	0,5662	0,4939	0,2271	0,0516
Weagle v1.2	0,5993	0,7451	0,5012	0,2159	0,0466

7 Related Work

Modeling a Web API through a folksonomy is, to the best of our knowledge, a new viewpoint that can be considered to support Web API selection and aggregation, given the multiple perspectives that can be assumed on Web API descriptions. In fact, different kinds of Web API and mashup models have been proposed in literature, focusing on a subset of Web API description elements, in order to serve different kinds of search, beyond the traditional category- or keyword-based ones.

Models for Web API Classification and Ranking. Some models are used to enable a more fine-grained classification of Web APIs based on their descriptive features (such as protocols and data formats) to go beyond the high level, error prone categorization given within public Web API repositories (for instance, through the 67 **ProgrammableWeb** categories). In [11] a classification of Web APIs is proposed by relying on the combination of traditional IR techniques (applied to Web API HTML documentation) and a taxonomy of APIs based on facets (e.g., **ProgrammableWeb** descriptive features). The paper introduces the Serviut score to rank Web APIs, computed within each category by considering

the Web API popularity according to Web traffic. In [8] the authors propose a quality model for Web APIs in order to rank them according to the quality of resources accessed through the Web APIs (in terms of accuracy, completeness, timeliness and availability) and the quality of the Web API interface in terms of usability and accessibility.

Models to Support Mashup Composition. These models focus on the definition of elements, such as Web API methods or GUI events, aiming at supporting mashup composition and automatic generation of Javascript gluing code between component Web APIs (e.g., with the help of a CASE tool). For instance, in [9] a component and a composition model are proposed to describe Web APIs and mashups for the development of dashboards; models are expressed by means of an XML-based language. In [1] a complex, formal model expressed through Datalog rules to capture all the aspects of mashups and their components is proposed. These models abstract from implementation issues, but often require a costly (typically manual) extraction and semantic annotation of this information from Web API documentation, although recent support tools to extract Web API models from available (unstructured) documentation have been proposed [16,19], in order to apply capability matching techniques that are similar to those already designed for web services [3].

Models for Web API Selection Based on Collective Knowledge. With *collective knowledge* we refer to the practice of relying on past experiences in using Web APIs in mashups to support their selection for new web applications under development [14,18]. Given a set of Web APIs, the system suggests combinations of Web APIs that have been jointly used in existing mashups. Starting from the occurrence of Web APIs within one or more mashups, approaches such as the ones described in [5,6,10,12] propose recommendation systems that exploit recurrent patterns of Web APIs within existing mashups. In [20,23] the same perspective is taken, but further focusing on the past use of other designers of the same Web APIs (social characterization).

Use of Folksonomies for Software Component Search. In [7] a novel way to model Web services using folksonomies is proposed. Instead of relying on complex Web service models taken from the Semantic Web tradition (such as OWL-S or WSMO), tags defined within a folksonomy are exploited to annotate service I/Os and tag hierarchies are automatically extracted. Nevertheless, in [7] the basic definition of folksonomy (see Definition 1) is adopted and only associative semantics techniques are used to infer sub-tag hierarchies. The same folksonomy model and very close techniques have been integrated within the Maracatu system [24] for search and retrieval of software components. The idea of extending the model for creating a Web API folksonomy, described here, has been sketched in [2]. Innovative contributions of this paper with respect to [2] concern the integration of techniques for building tag hierachies and the

definition of metrics based on the Web API folksonomy for supporting component search and aggregation.

8 Concluding Remarks

In this paper, we discussed the use of an innovative Web API folksonomy model for supporting component search and aggregation in the mashup development application context. First experimental evidences show promising results. The folksonomy of Web APIs has been modeled to be fully compliant with existing and commonly used public Web API repositories. It is not intended to substitute them, but to complement their contents in order to enable advanced Web API search facilities in such a collaborative environment. Further refinements on the folksonomy model we proposed in this paper might concern the use of the folksonomy itself as an abstraction from multiple Web API repositories (such as **ProgrammableWeb** or **Mashape**), the enrichment of the folksonomy by introducing social relationships among designers, the modeling of higher order aspects (for instance, designers may express opinions and assign ratings to other designers who used the Web APIs in their own mashups), the modeling of temporal aspects to be modeled in the folksonomy (for instance, how does the opinions of designers evolve in time).

References

1. Abiteboul, S., Greenshpan, O., Milo, T.: Modeling the Mashup Space. In: Proc. of the Workshop on Web Information and Data Management, pp. 87–94 (2008)
2. Bianchini, D.: Towards a folksonomy of web apis. In: Proceedings of the 3rd International Workshop on Semantic Search Over the Web (2013)
3. Bianchini, D., De Antonellis, V., Melchiori, M.: Capability matching and similarity reasoning in service discovery. In: CEUR Workshop Proceedings, vol. 160 (2005)
4. Bianchini, D., De Antonellis, V., Melchiori, M.: Semantic collaborative tagging for web aPIs sharing and reuse. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 76–90. Springer, Heidelberg (2012)
5. Bianchini, D., De Antonellis, V., Melchiori, M.: A linked data perspective for effective exploration of web APIs repositories. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 506–509. Springer, Heidelberg (2013)
6. Bianchini, D., De Antonellis, V., Melchiori, M.: A Multi-perspective Framework for Web API Search in Enterprise Mashup Design (Best Paper). In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 353–368. Springer, Heidelberg (2013)
7. Bouillet, E., Feblowitz, M., Feng, H., Liu, Z., Ranganathan, A., Riabov, A.: A Folksonomy-Based Model of Web Services for Discovery and Automatic Composition. In: IEEE Int. Conference on Services Computing, pp. 389–396 (2008)
8. Cappiello, C., Daniel, F., Matera, M.: A quality model for mashup components. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 236–250. Springer, Heidelberg (2009)
9. Cappiello, C., Matera, M., Picozzi, M., Sprega, G., Barbagallo, D., Francalanci, C.: DashMash: A Mashup Environment for End User Development. In: Auer, S., Díaz, O., Papadopoulos, G.A. (eds.) ICWE 2011. LNCS, vol. 6757, pp. 152–166. Springer, Heidelberg (2011)

10. Chowdhury, S., Chudnovskyy, O., Niederhausen, M., Pietschmann, S., Sharples, P., Gaedke, F.D.M.: Complementary assistance mechanisms for end user mashup composition. In: Proc. of the 22nd International Conference on World Wide Web Companion, pp. 269–272 (2013)
11. Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A., Verma, K.: A Faceted Classification Based Approach to Search and Rank Web APIs. In: Proc. of International Conference on Web Services (ICWS), pp. 177–184 (2008)
12. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: Proc. of the 35th Int. Conference on Very Large DataBases (VLDB), Lyon, France, pp. 538–549 (2009)
13. Grossman, D., Frieder, O.: Information Retrieval. Algorithms and Heuristics. Springer (2004)
14. Gruber, T.: Collective knowledge systems: Where the Social Web meets the Semantic Web. Journal Web Semantics: Science, Services and Agents on the World Wide Web 6, 4–13 (2008)
15. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: Search and ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
16. Maleshkova, M., Pedrinaci, C., Domingue, J.: Semantic annotation of Web APIs with SWEET. In: Proc. of the 6th Workshop on Scripting and Development for the Semantic Web (2010)
17. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating similarity measures for emergent semantics of social tagging. In: Proc. of the 18th Int. Conference on World Wide Web, pp. 641–650 (2009)
18. Montanelli, S., Bianchini, D., Aiello, C., Baldoni, R., Bolchini, C., Bonomi, S., Castano, S., Catarci, T., De Antonellis, V., Ferrara, A., Melchiori, M., Quintarelli, E., Scannapieco, M., Schreiber, F., Tanca, L.: The ESTEEM platform: Enabling P2P semantic collaboration through emerging collective knowledge. Journal of Intelligent Information Systems 36(2), 167–195 (2011)
19. Rodríguez, R., Espinosa, R., Bianchini, D., Garrigós, I., Mazón, J.-N., Zubcoff, J.J.: Extracting models from web API documentation. In: Grossniklaus, M., Wimmer, M. (eds.) ICWE Workshops 2012. LNCS, vol. 7703, pp. 134–145. Springer, Heidelberg (2012)
20. Shafiq, M., Alhajj, A., Rokne, J.: On the social aspects of personalized ranking for web services. In: Proc. of 13th IEEE Int. Conference on High Performance Computing and Communications, pp. 86–93 (2011)
21. Specia, L., Motta, E.: Integrating Folksonomies with the Semantic Web. In: Fraternali, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 624–639. Springer, Heidelberg (2007)
22. Spiliopoulos, V., Vouros, G., Karkaletsis, V.: On the discovery of subsumption relations for the alignment of ontologies. In: Web Semantics: Science, Services and Agents on the World Wide Web (2010)
23. Torres, R., Tapia, B., Astudillo, H.: Improving Web API Discovery by leveraging social information. In: Proceedings of the IEEE International Conference on Web Services, pp. 744–745 (2011)
24. Vanderlei, T.A., Durao, F.A., Martins, A.C., Garcia, V.C., Almeida, E.S., de L. Meira, S.R.: A Cooperative Classification Mechanism for Search and Retrieval Software Components. In: ACM Symposium and Applied Computing, pp. 866–871 (2007)

Adaptive Similarity of XML Data

Eva Jílková, Marek Polák, and Irena Holubová

Department of Software Engineering, Charles University in Prague, Czech Republic
`{polak,holubova}@ksi.mff.cuni.cz`

Abstract. In this work we explore application of XML schema similarity mapping in the area of conceptual modeling of XML schemas. We expand upon our previous efforts to map XML schemas to a common platform-independent schema using similarity evaluation based on exploitation of a decision tree. In particular, in this paper a more versatile method is implemented and the decision tree is trained using a large set of user-annotated mapping decision samples. Several variations of training that could improve the mapping results are proposed. The approach is implemented within a modeling and evolution management framework called *eXolutio* and its variations are evaluated using a wide range of experiments.

Keywords: XML schema matching, PSM-to-PIM mapping, model driven architecture.

1 Introduction

The XML (eXtensible Markup Language) [5] has become one of the leading formats for data representation and data exchange in the recent years. Due to its extensive usage, large amounts of XML data from various sources are available. Since it is common that sooner or later user requirements change, it is very useful to adapt independently created XML schemas that represent the same reality for common processing. However, such schemas may differ in structure or terminology. This leads us to the problem of *XML schema matching* that maps elements of XML schemas that correspond to each other. Schema matching is extensively researched and there exists a large amount of applications, such as data integration, e-business, schema integration, schema evolution and migration, data warehousing, database design and consolidation, biochemistry and bioinformatics, etc.

Matching a schema manually is a tedious, error-prone and expensive work. Therefore, automatic schema matching brings significant savings of manual effort and resources. But it is a difficult task because of the heterogeneity and imprecision of input data, as well as high subjectivity of matching decisions. Sometimes the correct matches have to be marked only by a domain expert. As a compromise, in semi-automatic schema matching the amount of user intervention is significantly minimized. For example, the user can provide information before matching/mapping, during the learning phase. Or, after creation of a

mapping (s)he can accept or refuse suggested mapping decisions which could be later reused for improvement of further matching.

In this work we apply the task of schema matching to a specific application of conceptual modeling – *MDA* (Model-Driven Architecture) [19]. MDA models the application domain at several levels of abstraction. *PSM* (Platform-Specific Model) schemas are integrated using a common conceptual schema – *PIM* (Platform-Independent Model) schema. In the optimal case, the PIM schema for a given domain is first designed and then various PSM schemas are derived from it for specific applications. In reality, the PIM schema has to be designed to describe a common domain in a situation when various PSM schemas for specific applications already exist. Or, a new PSM schema may need to be integrated with an existing hierarchy of PIM and PSM schemas. In both the cases independent PSM schemas may come from different sources, they may be of various types and they may use different naming conventions.

Schema matching is used as the key step during this integration process. In particular, we match elements from independent PSM schemas against elements in the common PIM schema to establish the respective PSM-to-PIM mapping. In particular, this work uses a semi-automatic approach to schema matching. We explore the applicability of decision trees for this specific use case. A decision tree is constructed from a large set of training samples and it is used for identification of correct mapping. For our target application various modifications of the training process are proposed and experimentally evaluated on the basis of several common hypotheses. The proposed approach extends our previous work [11] and it was implemented and experimentally tested in the modeling and evolution management tool called *eXolutio* [18] which is based on the idea of MDA.

The paper is structured as follows: In Section 2 we briefly describe the relation of schema matching and similarity and its usage in our target application. In Section 3 related work and existing implementations of schema matching are discussed. The proposed solution is described in Section 4. In Section 5 respective experiments are presented. Finally, results and possible future improvements are briefly resumed in Section 6.

2 Schema Matching

The semi-automatic or automatic process of finding correspondences between elements of two schemas is called *schema matching*. In this paper the term schema matching is used for simplicity in a general way, but there are various specific types. *Schema-to-schema matching* has as an input two XML schemas. *Instance-to-instance matching* has as an input two XML documents. And *schema-to-instance matching* has as an input an XML document and an XML schema. *Similarity* is a measure that expresses the level of correspondence. Its value is from interval $[0, 1]$, where 0 means no similarity and 1 means that the compared items are equal in the selected similarity meaning. *Matcher* is an algorithm that evaluates similarity of schemas according to particular criteria.

2.1 Usage of Schema Matching in MDA

Assume that an XSD¹ was created and we would like to integrate it now with a set of PSM schemas having a common PIM schema. Elements of the XSD are first converted to their corresponding PSM schema representatives. This conversion is straightforward, as it is proven in [20]. For the full integration we need to find the interpretation of its elements against the PIM elements. This could be done either manually or using schema matching. We explore usage of schema matching for this task in our work. A PSM element – PIM element pair is thus identified as an interpretation of PSM element against PIM element if it is suggested as a match by schema matching.

3 Related Work

In this section existing schema matching approaches are described. As we have mentioned, since the number of the approaches is high, we have selected only the key classical and most popular representatives.

3.1 COMA

COMA matcher [1] is an example of a *composite* approach. Individual matchers are selected from an extensible library of match algorithms. The process of matching is *interactive* and *iterative*. A match iteration has the following three phases: (1) *User feedback and selection of the match strategy*, (2) *Execution of individual matchers*, and (3) *Combination of the individual match results*.

Interactive Mode. The first step in the iteration is optional. The user is able to provide *feedback* (to confirm or reject previously proposed match candidates or to add new matches) and to define a *match strategy* (selection of matchers, strategies to combine individual match results). In *automatic mode* there is only one iteration and the match strategy is specified by input parameters.

Reuse of Match Results. Since many schemas to be matched are very similar to the previously matched schemas, match results (intermediate similarity results of individual matchers and user-confirmed results) are stored for later reuse.

Aggregation of Individual Matcher Results. Similarity values from individual matchers are aggregated to a combined similarity value. Several aggregate functions are available, for example *Min*, *Max* or *Average*.

Selection of Match Candidates. For each schema element its best match candidate from another schema is selected, i.e. the ones with the highest similarity value according to criteria like *MaxN* (n elements from schema S with maximal similarity are selected as match candidates), *MaxDelta* (an element from schema S with maximal similarity is determined as match candidate plus all S elements with a similarity differing at most by a tolerance value d which can

¹ XML Schema Definition.

be specified either as an absolute or relative value), or *Threshold* (all S elements showing a similarity exceeding a given threshold value t are selected).

The COMA++ [2], an extension of COMA, supports a number of other features like merging, saving and aggregating match results of two schemas.

3.2 Similarity Flooding

Similarity Flooding [4] can be used to match various data structures – data schemas, data instances or a combination of both. The algorithm is based on the idea that the similarity of an element is propagated to its neighbors. The input data is converted into *directed labeled graphs*. Every edge in the graphs is represented as a triple (s, l, t) , where s is a source node, t is a target node, and l is a label of the edge. The algorithm has the following steps: (1) *Conversion of input schemas to internal graph representation*, (2) *Creation of auxiliary data structures*, (3) *Computation of initial mapping*, (4) *Iterative fix-point computation* and (5) *Selection of relevant match candidates*. The accuracy of the algorithm is calculated as the number of needed adjustments. Output mapping of elements is checked and if necessary, corrected by the user.

Matcher. The main matcher is structural and is used in a *hybrid* combination with a simple name matcher that compares common affixes for initial mapping. The matcher is iterative and based on *fixpoint computation* with *initial mapping* as a starting point.

Fixpoint Computation. The similarity flooding algorithm is based on an iterative computation of σ -values. The computation of the σ -values for a map pair (x, y) is performed iteratively until the Euclidean length of the residual vector $\Delta(\sigma^n, \sigma^{n-1})$ becomes less than ϵ for some $n > 0$ (i.e. the similarities stabilize):

$$\sigma^{i+1}(x, y) = \sigma^i(x, y) + \sum_{\substack{(a, l, x) \in E_A \\ (b, l, y) \in E_B}} \sigma^i(a, b) w((a, b), (x, y)) + \sum_{\substack{(x, l, c) \in E_A \\ (y, l, d) \in E_B}} \sigma^i(c, d) w((x, y), (c, d)) \quad (1)$$

where $\sigma^i(x, y)$ is the similarity value in i -th iteration of nodes x and y and σ^0 is the value computed in the initial mapping.

Similarity Flooding can be further improved for example by usage of another matcher for initial mapping or auxiliary source of information – e.g. dictionary.

3.3 Decision Tree

In [3] a new method of combining independent matchers was introduced. It is based on the term *decision tree*.

Definition 1. A decision tree is a tree $G = (V, E)$, where V_i is the set of internal nodes (independent match algorithms), V_l is the set of leaf nodes (output decision whether elements do or do not match), $V = V_i \cup V_l$ is the set of all nodes, E is the set of edges (conditions that decide to which child node the computation will continue).

The decision tree approach does not have the following disadvantages of aggregation of result of independent matchers used, e.g., in COMA:

- **Performance:** In the composite approach with an aggregate function, all of the match algorithms have to run. The time required is worse than with a decision tree.
- **Quality:** Aggregation can lower the match quality, e.g., if we give higher weights to several matchers of the same type that falsely return a high similarity value.
- **Extendability** is worse, because adding a new matcher means updating the aggregation function.
- **Flexibility** is limited, because an aggregation function needs manual tuning of weights and thresholds.
- **Common Threshold:** Each match algorithm has its own value distribution, thus it should have own threshold.

The main disadvantage of Decision Trees is a need of a set of training data.

3.4 Advantages and Disadvantages

A general comparison of the previously discussed methods is introduced in Table 1. The decision tree approach seems to be the most promising for our application – it is dynamic and versatile. Furthermore it has desirable values of compared properties – it is highly extensible, quick and has a low level of user intervention and a low level of required auxiliary information.

Table 1. A comparison of the selected existing solutions

	Extensibility	Speed	User intervention	Auxiliary info
COMA	Low	Low	High	Low
Similarity Flooding	None	High	None	None
Decision Tree	High	High	Low	Low

4 Proposed Solution

First, we will briefly describe the algorithm for construction of decision tree proposed in [11] which we used for PSM-to-PIM mapping in our preliminary implementation called *eXolutio* [18] (as described later in Section 5). Then we will follow with a description of *C5.0* algorithm [16] that we utilized in our work for training of the decision tree. As we will show, it solves several problems of the original algorithm.

4.1 Original Decision Tree Construction

The decision tree in [11] is constructed as follows: The matchers are split into three groups (called *feature groups*) according to the main feature that they compare: *class name* (if the matcher compares names of the model classes), *data*

type (if the matcher compares similarity of data types of the given elements) and *structural similarity* (if the similarity is measured by the analysis of the models structures – relations of the nodes). In each feature group the matchers are assigned with a priority according to their efficiency. Then the matchers are sorted in ascending order according to importance of group (where for example in our case the class name group is the most important one) and their priority inside the group. Finally, the decision tree is built. The first matcher is selected as the root of the tree and other matchers are taken in sequence and added to the tree. If we want to add matcher M to the actual node n (i.e. use function $\text{addMatcherToTree}(M, n)$), there are the following possible situations:

- If node n has no child, method M is added as a child of n .
- If node n has children c_1, \dots, c_n from the same feature group that M belongs to and it has the same priority, then matcher M is added as the next child of node n .
- If node n has children c_1, \dots, c_n from the different feature group that M belongs to or it has a different priority, then for each node $c_i; i \in (1, n)$ we call $\text{addMatcherToTree}(M, c_i)$.

Though we have used the algorithm as the preliminary approach in our implementation, it has several drawbacks. First, it does not propose a method for automatic determination of conditions on edges and thresholds for continuous matchers. They have to be either set by the user or the default values are used. Furthermore, the decision tree does not suggest mapping results automatically. It computes an aggregated similarity score. During the traversal of the decision tree for each of the feature groups the maximal similarity value returned by the matcher from this group is stored. Then the aggregated similarity score is computed as an average of the maximal similarity value for each of the feature groups. For each PSM element it returns possible match candidates – PIM elements evaluated by the aggregate similarity score sorted in the descending order. This helps to find matches, but it is not done automatically – the user has to evaluate each mapping.

In this work we decided to generate the decision tree using machine learning techniques. This approach solves the above mentioned problems, as we would like to use the advantages of the decision tree approach and minimize the previous disadvantages.

4.2 Decision Tree Training via C5.0

Currently, there are several algorithms for the induction of a decision tree from training data, such as *ID3* [6], *CLS* [9], *CART* [8], *C4.5* [7], or *SLIQ* [10], to name just a few. The *C5.0* [16] algorithm is exploited and utilized in this paper for training of the decision tree. We decided to utilize for our purposes the *C5.0* because this algorithm and its predecessors are widely used and implemented in various tools, e.g. Weka [22]. First, we introduce a notation that is used in the rest of the section.

- S – a set of training samples. (An example of training samples is depicted in Figure 1.)
- $S(v, M)$ – a set of examples from S that have value v for matcher M .
- $S((i_1, i_2), M)$ – a set of examples from S that have value from interval (i_1, i_2) for matcher M .
- $C = \{C_1, C_2\}$ – the decision tree algorithm classifies S into two subsets with possible outcomes $C_1 = \text{match}$ and $C_2 = \text{mismatch}$.
- $\text{Info}(S)$ – entropy of the set S .
- $\text{freq}(C_i, S)$ – the number of examples in S that belong to class C_i .
- $|S|$ – the number of samples in the set S .
- $\text{Gain}(M, S)$ – the value of information gain for matcher M and set of samples S .
- $\text{Info}_M(S)$ – entropy for matcher M .

The entropy of the set of training samples S is computed as follows:

$$\text{Info}(S) = - \sum_{i=1}^2 \left(\frac{\text{freq}(C_i, S)}{|S|} \log_2 \left(\frac{\text{freq}(C_i, S)}{|S|} \right) \right) \quad (2)$$

The set S has to be partitioned in accordance with the outcome of matcher M . There are two possibilities:

1. Matcher M has n discrete values. In that case the entropy for matcher M and set S is computed as follows (using the above defined notation):

$$\text{Info}_M(S) = \sum_{i=1}^n \left(\frac{|S(i, M)|}{|S|} \text{Info}(S(i, M)) \right) \quad (3)$$

2. Matcher M has values from continuous interval $[a, b]$, that is why threshold $t \in [a, b]$ that brings the most information gain has to be selected by Algorithm 1. Entropy for matcher M and set S is then computed according to the following formulae:

$$\text{Info}_M(S) = \left(\frac{|S([a, t], M)|}{|S|} \text{Info}([a, t], M) \right) + \left(\frac{|S((t, b], M)|}{|S|} \text{Info}((t, b], M) \right) \quad (4)$$

The gain value for a set of samples S and matcher M is computed as follows:

$$\text{Gain}(M, S) = \text{Info}(S) - \text{Info}_M(S) \quad (5)$$

Then the decision tree is constructed by Algorithm 2 (which differs from the algorithm described in Section 4.1). There are the following possibilities for the content of the set of training samples S in the given node *parent* of the decision tree:

1. If S is empty, then the decision tree is a leaf identifying class C_i – the most frequent class at the parent of the given node *parent*. This leaf is added as a child to node *parent*.

2. If S contains only examples from one class C_i , then the decision tree is a leaf identifying class C_i . This leaf is added as a child to node $parent$.
3. If S contains examples from different classes, then S has to be divided into subsets. Matcher M with the highest value of information gain is selected. There are two possibilities:
 - (a) If matcher M has n discrete mutually exclusive values v_1, \dots, v_n , then set S is partitioned into subsets S_i where S_i contains samples with value v_i for matcher M .
 - (b) If matcher M has values (v_1, \dots, v_n) from continuous interval $[a, b]$, then threshold $t \in [a, b]$ has to be determined. Subsets S_1, S_2 contain samples with values from interval $[a, t], [t, b]$, respectively, for matcher M .

Matcher M is added as a child to node $parent$. For all the subsets S_i subtrees are constructed and added to node M as children.

The threshold for matcher M with values (v_1, \dots, v_n) from continuous interval $[a, b]$ is selected as follows:

- Values are sorted in the ascending order, duplicates are removed. Let us denote them u_1, \dots, u_m .
- All possible thresholds $A_i \in [u_i, u_{i+1}]$ have to be explored.
- For each interval $[u_i, u_{i+1}]$ the midpoint A_i is chosen as a split to two subsets $[u_1, A_i]$ and $(A_i, u_m]$.
- For each midpoint the information gain is computed and the midpoint A_{max} with the highest value of information gain is selected.
- The threshold is then returned as a lower bound of interval $[u_{max}, u_{max+1}]$.

Algorithm 1. Selection of threshold for continuous values v_1, \dots, v_n for matcher M and set of samples S

```

1: function COMPUTETHRESHOLD( $(v_1, \dots, v_n), S, M$ )
2:   /* sorts values in the ascending order, duplicate values are removed */
3:    $(u_1, \dots, u_m) \leftarrow \text{SORTASCDISTINCT}((v_1, \dots, v_n))$ 
4:   for  $i \leftarrow 1, m - 1$  do
5:     /* average of values  $U[i]$  and  $U[i+1]$  */
6:      $A[i] \leftarrow \text{AVG}(U[i], U[i + 1])$ 
7:      $L[i] \leftarrow U[i]$ 
8:      $H[i] \leftarrow U[i + 1]$ 
9:   end for
10:  /*  $u_m$  the highest value from the continuous interval,  $u_1$  the lowest value from the continuous
    interval */
11:  for  $i \leftarrow 1, m - 1$  do
12:     $gain_i \leftarrow \text{GAIN}(M, S([u_1, A[I]], (A[I], u_m], M))$ 
13:  end for
14:   $maxGain \leftarrow \max_{i=1}^{m-1} gain_i$ 
15:   $max \leftarrow i | gain_i = maxGain$ 
16:   $t \leftarrow L[max]$ 
17:   $threshCost \leftarrow \text{cost of splitting interval into two subintervals } [u_1, t] \text{ and } (t, u_m]$ 
18:   $result.gain \leftarrow maxGain - threshCost$ 
19:   $result.threshold \leftarrow t$ 
20:  return  $result$ 
21: end function

```

Algorithm 2. Construction of a decision tree T from a set S of user-evaluated training samples

```

1: function BUILDTREE( $S$ ,  $parent$ ,  $condition$ )
2:    $T$  empty tree
3:   if  $S$  is empty then
4:      $c \leftarrow$  the most frequent class at the parent of the given node  $parent$ 
5:     /* adds node  $c$  as a child to node  $parent$  with condition  $condition$  on edge */
6:     ADDLEAF( $parent$ ,  $c$ ,  $condition$ )
7:   else if  $S$  contains only results from one class  $C_i$  then
8:      $c \leftarrow C_i$ 
9:     /* adds node  $c$  as a child to node  $parent$  with condition  $condition$  on edge */
10:    ADDLEAF( $parent$ ,  $c$ ,  $condition$ )
11:   else
12:      $M \leftarrow$  matcher with the highest value of information gain  $Gain(M, S)$ 
13:     /* adds node  $M$  as a child to node  $parent$  with condition  $condition$  on edge */
14:     ADDNODE( $parent$ ,  $M$ ,  $condition$ )
15:     if  $M$  has  $n$  discrete mutually exclusive values  $v_1, \dots, v_n$  then
16:        $S' \leftarrow \{S_1, \dots, S_n\} | S_i = S(v_i, M)$ 
17:        $c_i \leftarrow v_i$ 
18:     else if  $M$  has values  $(v_1, \dots, v_n)$  from continuous interval  $[a, b]$  then
19:        $t \leftarrow COMPUTETHRESHOLD((v_1, \dots, v_n), M, S)$ 
20:       /* samples with values from interval  $[a, t]$  for matcher  $M$  */
21:        $S_1 \leftarrow S([a, t], M)$ 
22:        $c_1 \leftarrow [a, t]$ 
23:       /* samples with values from interval  $(t, b]$  for matcher  $M$  */
24:        $S_2 \leftarrow S((t, b], M)$ 
25:        $c_2 \leftarrow (t, b]$ 
26:        $S' \leftarrow \{S_1, S_2\}$ 
27:     end if
28:     for all  $S_i \in S'$  do
29:       /* constructs subtree  $T_i$  from subset  $S_i$  and adds it as a child to node  $M$  with
   condition  $c_i$  on edge */
30:        $T_i \leftarrow$  BUILDTREE( $S_i$ ,  $M$ ,  $c_i$ )
31:     end for
32:   end if
33:   return  $T$ 
34: end function
  
```

Example. To conclude, let us provide a simple illustrative example. For simplicity only three matchers are used: **Matched Thesauri** (which uses previous confirmed matching results for the evaluation), **Levenshtein Distance** (which computes the shortest edit distance from one string to another for operations insert, update and delete of a character) and **N-gram** (which computes the number of the same N-grams in two strings where an *N-gram* is a sequence of N characters in a given string). The C5.0 algorithm works in the following steps:

- In the beginning, the training set S contains 14 samples. **Matched Thesauri** has discrete values 0 and 1. **Levenshtein Distance** and **N-gram** have values from continuous interval $[0, 1]$. Gain values are computed for all matchers. **Matched Thesauri** has the highest gain value of 0.371, that is why **Matched Thesauri** is selected as the root of the constructed decision tree. Set S is divided into two parts $S(0, \text{Matched Thesauri})$ and $S(1, \text{Matched Thesauri})$.
- Set $S(1, \text{Matched Thesauri})$ contains samples that have value 1 for matcher **Matched Thesauri** and it contains only samples from the same match class. New leaf **match** is added as a child to node **Matched Thesauri**.

- Set $S_{MTO} = S(0, \text{Matched Thesauri})$ consists of samples with value 0 for matcher **Matched Thesauri** and results from various classes, so this set has to be further divided. Gain values are computed and matcher with the highest gain value, i.e. **N-gram**, is added as a child of node **Matched Thesauri**. The threshold value for **N-gram** matcher with continuous range is 0.071 and set S_{MTO} is divided into two subsets $S_{N1} = S([0, 0.071], \text{N-gram})$ and $S_{N2} = S((0.071, 1], \text{N-gram})$.
 - There are only mismatch results in set S_{N1} , so leaf **mismatch** is added as a child to node **N-gram**.
 - Set S_{N2} also contains results from one class – match. Another leaf **match** is added to node **N-gram**.

The final trained decision tree is displayed in Figure 2.

Matched Thesauri		Levenshtein Distance		N-gram				
Discrete		Continuous		Continuous				
1	0		0.167		0	match		
0	0		0.1		0	mism		
1	0.2			0.063	match			
1	0		0	0	match			
0	0		0	0	mism			
0	0.1			0	mism			
0	0.5			0.25	match			
0	0		0	0	mism			
0	0.1			0	mism			
0	0.429			0.25	match			
0	0.125			0.071	mism			
0	0.6			0.385	match			
1	0.2			0.067	match			
match	mism	0.183	match	mism	0.032	match	mism	
0	3	7	<=	2	7	<=	2	6
1	4	0	>	5	0	>	5	1
0.371		0.324		0.115				
Matched Thesauri = 0								
Matched Thesauri		Levenshtein Distance		N-gram				
Discrete		Continuous		Continuous				
0	0		0.167		0	mism		
0	0		0.1		0	mism		
0	0		0		0	mism		
0	0		0.1		0	mism		
0	0.5			0.25	match			
0	0		0	0	mism			
0	0.1			0	mism			
0	0.429			0.25	match			
0	0.125			0.071	mism			
0	0.6			0.385	match			
match	mism	0.298	match	mism	0.161	match	mism	
0	3	7	<=	0	7	<=	0	7
1	0	0	>	3	0	>	3	0
0		0.762		0.781				

Fig. 1. Data used for training of decision tree in example

5 Experiments

For the purpose of evaluation of the described approach we have performed an extensive set of experiments. Due to space limitations and complexity of the experiments, this section contains description of only one of the experiments and a discussion of its results. The complete set of experiments can be found

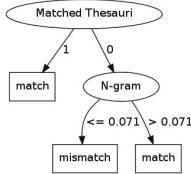


Fig. 2. Sample decision tree

in [21]. Particular experiments differ in used sets of matchers, training sets, etc.
 - efficiency of measure methods depending on element types; usage of different matchers; comparison of domain thesaurus and user-confirmed matches; quality of matching of decision trees trained from different sets.

The proposed approach was implemented in the *eXolutio* [18] tool and replaces the original approach [11] (whose disadvantages were described in Section 4.1). *eXolutio* is based on the MDA approach and models XML schemas at two levels – PIM and PSM. *eXolutio* allows the user to manually design a common PIM schema and multiple PSM schemas with interpretations against the PIM schema. Mapping between the two levels allows to propagate a change to all the related schemas.

All experiments were run on a standard personal computer with the following configuration: Intel(R) Core(TM) i5-3470 3.20 GHz processor, 8 GB RAM, OS 64-bit Windows 7 Home Premium SP1.

The following sets of XML schemas have been used for training of the decision tree:

- BMEcat is a standard for exchange of electronic product catalogues².
- OpenTransAll is a standard for business documents³.
- OTA focuses on the creation of electronic message structures for communication between the various systems in the global travel industry⁴.

PIM Schemas. A PIM schema used for experiments describes a common interface for planning various types of holidays. It can be found in [21].

XML Schemas for Experiments. For evaluation the following XML schemas were used:

- Artificial XML schema 01_Hotel designed for the purpose of this work. It describes basic information about hotels.
- Realistic XML schemas: 02_HotelReservation⁵, 03_HotelAvailabilityRQ⁶

² www.bmecat.org

³ www.opentrans.de

⁴ www.opentravel.org

⁵ <http://kusakd5am.mff.cuni.cz/hb/schema/reservation>

⁶ <http://itins4.madisoncollege.edu/IT/152121advweb/XMLExamples/unit3/schemaSimple/HotelAvailabilityRQ.xsd>

Domain Thesaurus. The domain thesaurus contains sets of words that are related semantically. The thesauri are used during matching by **Dictionary** matcher. The domain thesaurus for the domain of hotels is as follows (related words are marked by \sim): address \sim location, accommodation \sim hotel, boarding \sim meal, count \sim amount, lengthOfStay \sim numberOfNights.

5.1 Separate Decision Trees and Common Decision Tree for Classes and Attributes Experiment

The presented experiment is designed from the following observation: Efficiency of methods used to measure similarity between elements depends on the type of elements – if they are **classes** or if they are **attributes**. In this experiment two sets of decision tree are used: *two separate trees for classes and for attributes* and *one common tree for classes and attributes*.

Experiment Setup

- **Used Schemas:** 01_Hotel, 02_HotelReservation, 03_HotelAvailabilityRQ
- **Decision Tree:**
 - Separate decision tree for attributes (in Figure 3) and for classes (in Figure 4)
 - Common decision tree (in Figure 5)
- **Decision Tree Training Set:**
 - Set of XSD Schemas: OTA
 - Sample count:
 - * Separate decision tree for attributes: 27,942 match pairs
 - * Separate decision tree for classes: 27,793 match pairs
 - * Common decision tree: 55,815 match pairs
- **Thesaurus for Dictionary:** None
- **Thesaurus for Matched Thesauri:** None
- **Matchers:** Children (which compares the structural similarity of child nodes or neighboring nodes of classes), DataType (which compares data types), Dictionary, Length Ratio (which computes the ratio of lengths of two input strings), Levenshtein Distance, Matched Thesauri, Prefix (which compares whether the string s_1 is a prefix of the string s_2 or the other way around)

In the presented experiment no additional source of information was used. Both sets of decision trees were trained without domain thesauri and without previous user matches. Both sets of decision trees are induced from the same set of training samples OTA, particularly match pairs of XML schema OTA_HotelAvailGetRQ.xsd and XML schema OTA_HotelAvailGetRS.xsd. A separate decision tree for classes and attributes is trained only from match pairs of classes and attributes respectively. The common tree is trained from both sets together. The final decision trees are shown in Figures 3, 4 and 5.

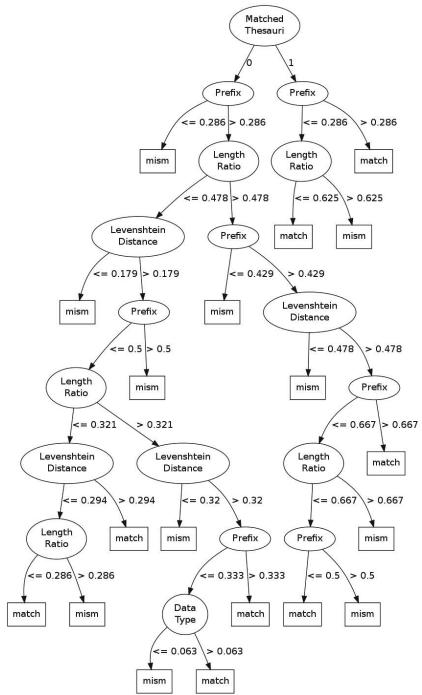


Fig. 3. A separate decision tree for attributes for experiment *SeparateTrees*

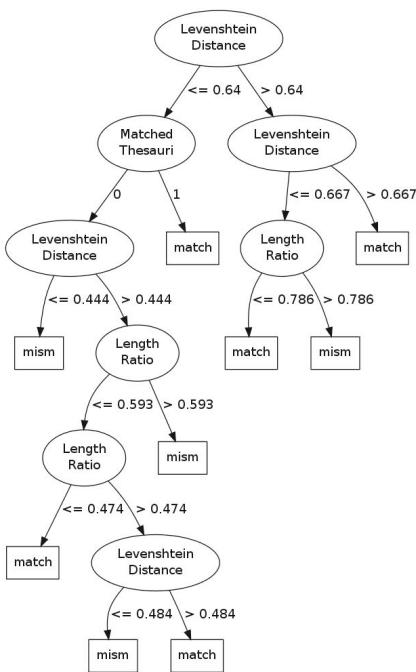


Fig. 4. Separate decision tree for classes for experiment *SeparateTrees*

The root of the separate decision tree for attributes is **Matched Thesauri**, all the other trees in this experiment have **Levenshtein Distance**. The matcher at the second level is the same for both branches and they have the same threshold. Especially the subtree for mapping pairs that are contained in **Matched Thesauri** is interesting. We would assume that this subtree should be smaller or even a leaf with the value ‘match’. This could be explained by errors in user annotation of mapping results – the same match pair is annotated with different matching decision than the previous one or some mapping pairs have different meaning in different context.

The separate decision tree for classes is relatively simple. It contains only matchers **Levenshtein Distance**, **Matched Thesauri** and **Length Ratio**, other matchers are not used. It corresponds with the original observation that some methods are more effective for certain types of elements. Matchers whose similarity values do not distinguish mapping pairs enough are not included. Pairs that are contained in the thesaurus are directly suggested as matches.

The threshold value for **Matched Thesauri** matcher in the root of the common tree and the separate tree for classes is nearly similar. The common decision tree



Fig. 5. Common decision tree for experiment *SeparateTrees*

is the most complex from the above mentioned. The common tree also contains two subtrees for **Matched Thesauri**. The first one is at the second level and it contains two full subtrees for both the values. The right subtree for pairs that are contained in thesauri is more complex than the tree in the separate tree for attributes. This could be caused by a larger number of training samples that allows for more detail resolution. The second one, i.e. **Matched Thesauri**, is directly a parent of the leaves.

In Figures 6, 7, 8 and 9 there are displayed the histograms of the match quality measures – Precision, Recall, F-Measure and Overall respectively. All

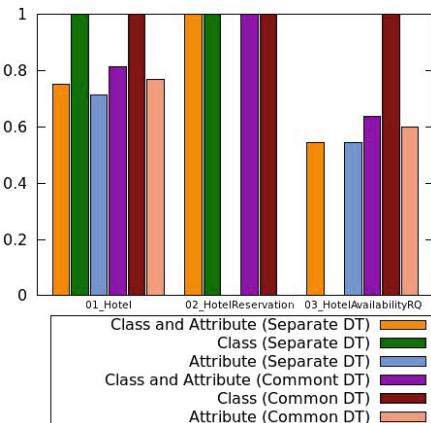


Fig. 6. Precision for experiment *Separate-Trees*

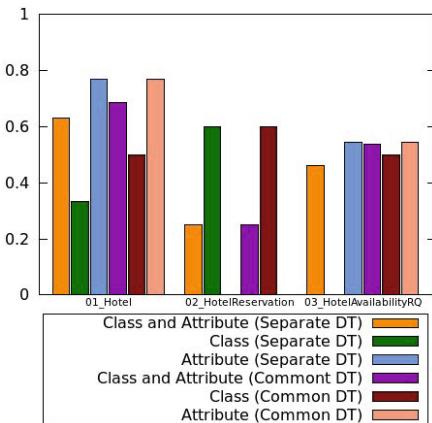


Fig. 7. Recall for experiment *Separate-Trees*

the measures are at first computed for both types of elements together and then separately for attribute and class elements.

In Figure 6 Precision is high for classes in all schemas and for both types of trees. The quality of mapping decision differs significantly with the type of element, but the training set contains a similar number of match pairs for classes (27,793 match pairs) and attributes (27,942 match pairs). The separate tree for classes did not suggest any mapping pair as a match for schema 03_HotelAvailabilityRQ, just as the separate tree for attributes for schema 02_HotelReservation. Attributes in schema 03_HotelAvailabilityRQ are difficult to identify for all the decision trees. All Precision values are from the interval [0.545, 0.769] – they identified almost the same number of relevant results as irrelevant.

Recall is lower than Precision in all the cases except for schema 01_Hotel and the separate tree for attributes in Figure 7. There were no true positives attributes for schema 02_HotelReservation for both trees and no true positives classes for schema 03_Hotel-AvailabilityRQ for separate tree. Values of Recall are lower for attributes than Recall for classes.

In Figure 8 the values for F-Measure are equal for schema 02_HotelReservation for classes for both trees. Post-match effort for adding false negatives (FN) and removing false positives (FP) is quite high in all cases in Figure 9. The highest value of Overall is 0.6.

The hypothesis was not confirmed, all similarity measures are higher for the common decision tree that is trained from a bigger set of training examples, the quality of the decision tree seems to depend more on the size of the set of training samples. The best score was achieved for Precision. Both trees in this experiment had a larger number of FN than FP. They miss a match suggestion

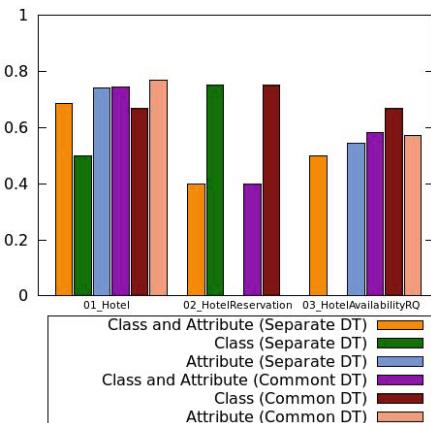


Fig. 8. F-Measure for experiment *SeparateTrees*

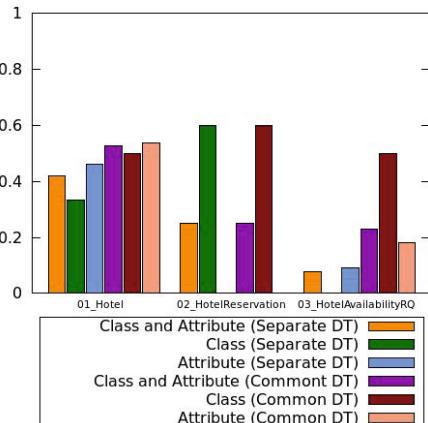


Fig. 9. Overall for experiment *SeparateTrees*

more than they incorrectly suggest it as a match pair. It could be improved by adding an auxiliary source of information or a new matcher.

Examples of matching results from this experiment are shown in Table 2. Match pair **numberOfNights** – **LengthOfStay** is difficult to identify without an auxiliary source of information for both sets of decision tree. Match pairs **CheckOutDate** – **CheckOut**, **ContactInfo** – **Contact** and **BedType** – **ReservationType** were identified correctly by the common tree and incorrectly by separate decision trees.

Table 2. Examples of mapping results for experiment *SeparateTrees*

XSD	PIM	DT type	DT	User	Result
C ContactInfo	Contact	Separate	Mismatch	Match	FN
C ContactInfo	Contact	Common	Match	Match	TP
A Fax	FaxNumber	Separate	Match	Match	TP
A Fax	FaxNumber	Common	Match	Match	TP
A numberOfNights	LengthOfStay	Separate	Mismatch	Match	FN
A numberOfNights	LengthOfStay	Common	Mismatch	Match	FN
A CheckOutDate	CheckOut	Separate	Mismatch	Match	FN
A CheckOutDate	CheckOut	Common	Match	Match	TP
A BedType	ReservationType	Separate	Match	Mismatch	FP
A BedType	ReservationType	Common	Mismatch	Mismatch	TN

Further experiments with various hypotheses, e.g. training with sets of different sizes, with different sets of matchers, or with usage of auxiliary information, can be found in [21].

6 Conclusion

Schema matching, i.e. the problem of finding correspondences, relations or mappings between elements of two schemas, has been extensively researched and has a lot of different applications. In this paper a particular application of schema matching in MDA is explored. We have implemented our approach within a modeling and evolution management tool called *eXolutio* [18] which is based on the idea of MDA. This mapping is very useful in case of a change, because changes in one place are propagated to all the related schemas. The presented schema matching approach is used to identify mappings between PIM and PSM level of MDA, representing an interpretation of a PSM element against a PIM element.

We have explored various approaches to schema matching and selected the most promising possible approach for our application – schema matching using a decision tree. This solution is dynamic, versatile, highly extensible, quick and has a low level of user intervention and a low level of required auxiliary information. We have extended the previous work [11] by utilization of the C5.0 algorithm for training of decision tree from a large set of user-annotated schema pairs. Our approach is now more versatile, extensible and reusable. Further we evaluated our approach on a wide range of experiments and implemented a module that is easily extensible. We also implemented a user-friendly interface for evaluation of mappings suggested by the decision tree, i.e. a solid background for further experiments.

A straightforward extension of this work is to expand the set of available matchers with more powerful matchers, e.g. with a matcher that uses the *WordNet*⁷ thesaurus for synonyms. Further possibilities are for example string matchers⁸ and the *Soundex* matcher⁹. Also the user interface leaves a space for improvement. We could add an interface for evaluation of matches during the preparation phase of decision tree training or dynamic editing of trained decision tree – remove, move, add matcher node, change results in leaves or threshold on edges.

Acknowledgment. This work was supported by the project SVV-2014-260100 and the GAUK grant no. 1416213.

References

1. Do, H.H., Rahm, E.: COMA – A system for flexible combination of schema matching approaches. In: Proceedings of the 28th International Conference on Very Large Data Bases, Pages, pp. 610–621. VLDB Endowment, Hong Kong (2002)
2. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and Ontology Matching with COMA++. In: Proceeding SIGMOD 2005 Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 906–908 (2005) ISBN:1-59593-060-4

⁷ <http://wordnet.princeton.edu/>

⁸ <http://secondstring.sourceforge.net/>

⁹ <http://www.archives.gov/research/census/soundex.html>

3. Duchateau, F., Bellahsene, Z., Coletta, R.: A flexible approach for planning schema matching algorithms. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 249–264. Springer, Heidelberg (2008)
4. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm. In: Proceeding ICDE 2002 Proceedings of the 18th International Conference on Data Engineering, p. 117. IEEE Computer Society, Washington, DC (2002)
5. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0, 5th edn. W3C Recommendation (November 26, 2008), <http://www.w3.org/TR/REC-xml>.
6. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1(1), 81–106 (1986)
7. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc, San Francisco (1993) ISBN:1-55860-238-0
8. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Chapman & Hall, New York (1984)
9. Hunt, E. B., Marin, J., Stone, P. T.: Experiments in Induction. Academic Press, New York (1966)
10. Mehta, M., Agrawal, R., Rissanen, J.: SLIQ: A fast scalable classifier for data mining. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–540. Springer, Heidelberg (1996)
11. Stárka, J.: Similarity of XML Data. Master's thesis, Charles University in Prague (2010), <http://www.ksi.mff.cuni.cz/~holubova/dp/Starka.pdf>
12. Shasha, D., Zhang, K.: Approximate Tree Pattern Matching. Pattern Matching Algorithms, pp. 341–371. Oxford University Press (1997)
13. Nierman, A., Jagadish, H.V.: Evaluating Structural Similarity in XML Documents. In: Proceedings of the Fifth International Workshop on the Web and Databases, pp. 61–66 (2002)
14. Li, W., Clifton, C.: SemInt: a tool for identifying attribute correspondences in heterogeneous databases using neural network. Data & Knowledge Engineering 33(1), 169–123 (2000) ISSN 0169-023X
15. Chen, P.: The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems, 9–36 (March 1976)
16. Quinlan, R.: C5.0, <http://www.rulequest.com/see5-unix.html>.
17. Stárka, J., Mlýnková, I., Klímek, J., Nečaský, M.: Integration of web service interfaces via decision trees. In: Proceedings of the 7th International Symposium on Innovations in Information Technology, pp. 47–52. IEEE Computer Society, Abu Dhabi (2011) ISBN: 978-1-4577-0311-9
18. Klímek, J., Mlýnková, I., Nečaský, M.: eXolutio: Tool for XML and Data Management. In: CEUR Workshop Proceedings, pp. 1613–1673 (2012) ISSN: 1613-0073
19. Miller, J., Mukerji, J.: MDA Guide Version 1.0.1. Object Management Group (2003), <http://www.omg.org/docs/omg/03-06-01.pdf>
20. Nečaský, M., Mlýnková, I., Klímek, J., Mařík, J.: When conceptual model meets grammar: A dual approach to XML data modeling. International Journal on Data & Knowledge Engineering 72, 1–30 (2012) ISBN:3-642-17615-1, 978-3-642-17615-9
21. Jílková, E.: Adaptive Similarity of XML Data. Master's thesis, Charles University in Prague (2013), <http://www.ksi.mff.cuni.cz/~holubova/dp/Jilkova.pdf>
22. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explorations 11(1) (2009)

FAGI: A Framework for Fusing Geospatial RDF Data

Giorgos Giannopoulos¹, Dimitrios Skoutas¹, Thomas Maroulis²,
Nikos Karagiannakis¹, and Spiros Athanasiou¹

¹ IMIS Institute, “Athena” Research Center

² Imperial College, London

Abstract. In this paper, we present FAGI, a framework for fusing geospatial RDF data. Starting from two interlinked datasets, FAGI handles all the steps of the fusion process, producing an integrated, richer dataset that combines entities and attributes from both initial ones. In contrast to existing approaches and tools, which deal either with RDF fusion or with spatial conflation, FAGI specifically addresses the fusion of geospatial RDF data. We describe the main components of the framework and their functionalities, which include aligning dataset vocabularies, processing geospatial features, applying -manually or automatically- fusion strategies, and recommending link creation or rejection between RDF entities, with emphasis on their geospatial properties.

1 Introduction

Languages and standards for organizing and querying semantic information, such as RDF(S) and SPARQL, are increasingly being adopted not only within academic communities but also by corporate vendors, which turn to semantic technologies to more effectively organize, expose and exchange their data as Linked Data. However, it is often the case that different data sources, although describing the same real world entities, provide different views of them, either by providing information on different subsets of attributes or even by providing different values on the same attributes. Typical reasons for this is that some sources may be outdated or may serve a different purposes. For example, different maps a city’s roads and buildings, obtained from different sources (e.g., governmental, commercial, crowdsourced), may differ in the geometries and coordinates of the depicted geospatial features, as well as on the type, richness and correctness of the metadata associated with them (e.g. names and categories of buildings). As a result, information for the same real world entities is often spread across several heterogeneous datasets, each one providing partial and/or contradicting views of it, which then need to be fused in order to acquire a richer, cleaner and universal dataset.

This fusion task constitutes the final main part of the data integration process, following the steps of schema integration and record linkage [2]. Fusion handles the merging of the linked entities, that is the production, for each set of linked entities, of a richer, more correct and more complete description, w.r.t. to the attributes describing it. Fusion involves recognizing which attributes/properties of the entities correspond to each other and resolving potential conflicts or irregularities, such as different values for the same properties, lack of values or properties, differences in metadata quality, etc.

In this paper, we focus on fusion of geospatial RDF data. We first examine the existing approaches in the two individual fields of fusion of RDF data and fusion of

geospatial data to identify open problems and challenges. Based on our findings on the shortcomings of previous works, we propose a fusion framework called FAGI (Fusion and Aggregation for Geospatial Information) that includes and accommodates all the aspects of the process of fusing geospatial RDF data, dealing with the open issues and specificities of this case.

The rest of the paper is organized as follows. Sections 2 and 3 review state-of-the-art techniques for RDF data fusion and spatial conflation, respectively. Then, the overall approach and challenges are presented in Section 4. Section 5 outlines the FAGI architecture and its main components. Finally, Section 6 concludes the paper.

2 RDF Data Fusion

Although several works exist on schema integration and interlinking of RDF data, fusion has received less attention and is still a field of ongoing research. Existing approaches typically adopt existing fusion concepts and do not provide ways to automate the fusion process. Below, we describe the main existing fusion tools for RDF data.

Sieve [10] focuses on quality assessment and fusion of Linked Data, constituting part of a larger framework for Linked Data integration [18] that provides state-of-the-art techniques for data fusion [1]. The results of the quality assessment process are utilized by the fusion process. Fusion takes into account factors such as timeliness of data, provenance, as well as user configurable preference lists on features of the dataset. In particular, Sieve supports the following strategies: (i) *Filter*: removes all values for which the input quality assessment metric is below a given threshold; (ii) *KeepFirst*: keeps the value with the highest score for a given quality assessment metric; (iii) *KeepAllValuesByQualityScore*: similar to *KeepFirst*, but in case of ties, it keeps all values with the highest score; (iv) *Average*: takes the average of all input data for a given property; (v) *Voting*: picks the value that appeared most frequently across sources; (vi) *WeightedVoting*: picks the most frequent value across highly rated sources.

ODCleanStore [8,9] is another framework that includes fusion functionality of RDF data. It supports linking, cleaning, transformation, and quality assessment operations on Linked Data. The fusion component supports several user configurable conflict resolution strategies for fusion, that also consider provenance and quality metadata of the datasets: (i) *ANY, MIN, MAX, SHORTEST, LONGEST*: an arbitrary value, minimum, maximum, shortest, or longest is selected, respectively, from the conflicting values; (ii) *AVG, MEDIAN, CONCAT*: computes the average, median, or concatenation, respectively; (iii) *BEST*: the value with the highest aggregate quality is selected; (iv) *LATEST*: the value with the newest time is selected; (v) *ALL*: all input values are kept.

KnoFuss [12] is a framework for interlinking, conflict detection, and fusion, with main focus on interlinking (coreferencing). It implements several variations of the Jaro-Winkler string similarity metric and an adaptive learning clustering algorithm, which are applied in a way, configurable to the ontology classes of the respective entities.

ALOE [11] handles automatic fusion of RDF data. It supports automatic discovery of class and property mappings across endpoints even when no schema information is available. Moreover, it applies machine learning techniques to learn transformation/fusion rules for string values at several levels (i.e. characters, n-grams, words). It

offers two learning modes: batch and active learning. ALOE is currently still under development; however, experiments performed on its effectiveness [11] yield encouraging results w.r.t. the task of automatically fusing textual RDF properties.

3 Geospatial Data Fusion

In the domain of geospatial data management, fusion or conflation is the process of merging two different datasets that cover overlapping regions, in such way that the best quality elements of the individual datasets are kept in the final, composite dataset [4]. The conflation process may be applied to any combination of vector (i.e. points, polygons) and raster (i.e. images) data. Next, we focus only on fusion approaches applicable to vector data. Geospatial data fusion has been an area of study for several decades; hence, a considerable amount of approaches handling the problem exist.

One of the earliest works in the field proposes an automatic, iterative process for conflating geospatial objects [17]. It iteratively identifies new matching points (control point pairs) and transforms geometries, until no new point pairs can be identified. More specifically, the process comprises the following steps: (i) Test criteria (e.g. distance, direction, non-spatial metadata) are defined to identify matching point pairs from the two datasets. (ii) Test criteria are combined to perform matching. This is handled specially at the first iteration, since the purpose then is to identify the most accurate point pairs. So, the focus is on strict matching based on geospatial proximity. (iii) Criteria are applied to produce matching point pairs. (iv) Matching points are aligned by first partitioning the space through triangulation ([14]) based on the matching points and then transforming each partition with rubber-sheeting, which essentially stretches the space of the two datasets until they are aligned. (v) The process is repeated starting from step 2, until no more matching point pairs can be identified. The evaluation of the method demonstrates fairly good accuracy of matching, pointing out the trade-off between false-positive and false-negative results.

A multi-step, but not iterative, process is also proposed in [20] for matching spatial objects. It focuses on fusing streets, utilizing several statistics of the spatial objects as matching criteria. Briefly, the steps of the process are the following: (i) preprocessing, which incorporates control points identification and space transformation; (ii) identification of potential matching geometry pairs, where buffers are created around candidate matching elements to guide the matching; (iii) filtering through geometric constraints application (iv) evaluation of matching pairs, where the list of matching pairs is further refined with a merit function that utilizes statistics of the dataset; and (v) calculation of the final matchings, where the search space is partitioned into smaller parts, so that the matchings of the objects can be efficiently evaluated. The method produces satisfying matching results, but it requires prohibitively large execution times.

A similar approach is adopted in [19], using buffers around spatial objects to examine containment relations between objects (edges between points). The process comprises feature matching and feature updating. During feature matching, if necessary, edges are divided in such a way that full containment between edges and/or edge parts is achieved. During updating, the matched edges are replaced with weighted averages and are combined with edges of the datasets that could not be matched at the first step.

The approach proposed in [3] is based on the Vector Product Format (VPF), a standard for organizing geospatial data, including thematic information (i.e. categories of spatial objects), as well as topological relations between them (e.g. neighbouring objects). Feature matching is performed by combining spatial and semantic criteria, i.e. both geospatial proximity and similarity of feature descriptions and categories. For feature alignment, triangulation and rubber-sheeting techniques are applied. A final de-confliction step is applied, where a unified feature is produced out of the individual matching features, using quality metadata stored in the VPF model.

A more recent approach is described in [5], focusing on conflating road networks. Road intersections are utilized as control points, based on previous works indicating that intersection points are good candidates for feature matching. Then, for each of the datasets to be fused, the distribution of the intersection points is analysed, and feature matching is performed on intersection point sets. Several optimizations are applied to reduce the number of point sets to be compared, utilizing distances, angles and node degrees of intersection points. The method achieves high precision and recall values.

In [16], the problem of geospatial data fusion is discussed as a means to increase data quality. The main focus is on improving the quality of road networks by fusing roundabouts. To this end, the degrees of the points of the roundabout and the angles between the road segments are exploited. The same authors present also algorithms as well as a tool for fusing geospatial features [15]. The main focus is first on data preprocessing (decomposition of polyline based datasets and extraction of branches) and merging of geometrical and semantic information (branch assignment, integration of geodata). The evaluation shows high accuracy; however, the authors point out some shortcomings of the methods, such as applicability only on restricted search spaces.

Summarizing, a common ground in several proposed methods is the identification of control point pairs, mainly based on strict geospatial matching (utilizing distances, angles, lengths, etc.). Another commonly adopted practice is the partition of the space into sub-areas, usually applying triangulation based on the control points. This serves as a pre-processing step so that a space transformation is applied to the datasets to align the rest of the geometries to each other, according to the alignment of the control points. Some of the proposed methods utilize a combination of spatial and non-spatial criteria in matching functions, while others use statistics on geospatial properties and relations in the datasets; moreover, certain methods are iterative, while others not.

A drawback of most methods seems to be their high execution time. This is mainly due to the very complex geospatial operations that are required in several steps of the process, such as normalization, calculation of spatial relations, space transformations, etc. This is also due to the large search space, which is the number of comparisons to be made between objects of the two compared datasets. To this end, partition, clustering or candidate items filtering techniques are applied, without, however, significantly improving the efficiency of the methods. Another common drawback is also the limited utilization of semantic, non-spatial properties of the geospatial features, which can contribute both by increasing the accuracy of the feature matching process and by reducing the search space. This results from the fact that most methods in the literature are designed for processing conventional geospatial data, lacking the semantic richness of geospatial RDF datasets.

4 FAGI Approach and Challenges

The problem of fusing geospatial RDF datasets presents some specific challenges that differentiate it from the individual settings of fusing geospatial data or fusing RDF data, since both semantic and spatial knowledge is available and needs to be taken into account during the fusion process. In the case of RDF data, there is a clear and explicit separation between the two steps of interlinking and fusion: first, matching entities from the two input datasets are identified and linked together; then, their attributes are fused to produce a single entity representation. In the case of geospatial data, matching and fusing are rather intertwined. Hence, when dealing with RDF entity pairs from two compared datasets, the additional knowledge that these pairs correspond to the same real world object, and thus, their geometries correspond to each other, is now given. Still, the fusion process in this case needs to handle more complex sets of attributes, including strings or arithmetic values in conjunction with geometries of entities, which in certain scenarios requires transformations/combinations of existing geometries.

To address these specificities, a set of fusion strategies and actions are needed to support the operations described below:

- *Define effective scoring functions that meaningfully combine semantic properties and geospatial features.* These functions should utilize: (a) available mappings on the schemata that characterize the interlinked entities; (b) refined similarity metrics individually fitting different attributes of the entities (e.g., names and descriptions).
- *Define quality criteria to select among different geometries.* For example, in some cases the most accurate geometry might be considered to be the most complex one or the most recent one. In other cases, choosing the most accurate geometry may be possible based on available quality indicators of the datasets.
- *Combine geometries to produce a richer geometry.* In some cases it may not be sufficient to simply keep one of the two initial geometries, but it may be needed instead to combine parts of them to produce a more accurate geometry for the entity. This can be based on various strategies. One possibility is to keep partial geometric properties from both geometries, to produce a new geometry of higher quality. For example, in an interlinked pair of geometries, if geometry A is a point with high quality coordinates and the other geometry is a polygon B centered around a different point, then the fused geometry could be the polygon B shifted so that its new centroid is point A. In other cases, it may be desired to keep both geometries separately. For example, the geometries could indeed correspond to the same real world entity but representing parts of it (e.g. the building of a school and its yard).
- *Validating and updating links between the two datasets.* Comparing the geometries of the interlinked entities can be used as further evidence for validating the links between entities of the two datasets (if this was not already considered as a criterion during the linking process). More specifically, when the matching score of two interlinked geometries is too low, this can be an indication that these two geometries have been falsely interlinked. Moreover, new links between entities could be identified after examining clusters/neighborhoods of other, already linked entities and applying traditional geospatial conflation techniques (space transformation, statistics on spatial properties). This makes the problem even more challenging, since it

makes the process iterative, increasing both the complexity of specifying the rules for the fusion process and its computational complexity.

- *Apart from fusing the attributes of the matched entities, produce an overall fused dataset that contains entities from both initial datasets.* This entails a process similar to the one described above: by searching a neighborhoods of interlinked entities, and concluding that an entity of dataset B does not exist in dataset A, the entity can be created in A too, and be interlinked with the respective entity of dataset B.
- *Produce new semantic, topological relations.* Based on analysis of the geospatial relations between interlinked entities, some basic geospatial relations between entities could be calculated and stored as semantic metadata of the entities. For example, properties that capture, for each entity, the orientation (e.g. north from) of its top-k closest neighbors could be created.

A further difference is that, instead of searching the whole datasets to select control points, the interlinked entities can now be utilized as control point pairs for partitioning the space and working on each partition separately.

Finally, an important challenge is to implement effective learning algorithms that will be able to train on user actions in order to recommend both fusion scores and preferred fusion strategies. A critical factor for the effectiveness of such algorithms is the proper selection of training features, i.e. features that describe interlinked pairs of entities/geometries and correlate them to the fusion strategy selected by that user. Moreover, algorithms are needed for both offline and online modes; that is, the algorithms should be able to be trained either offline, on previous user actions, or online, as the user manually selects a fusion action for a set of interlinked entity pairs.

5 FAGI Architecture and Main Components

Based on the above, we describe the proposed framework for fusing geospatial RDF data, called FAGI, and its main components. FAGI aims at providing a comprehensive solution for geospatial enrichment of knowledge bases, based on semantic technologies and open data sources. Its main functionality is to support the entire process of aligning RDF representations of geospatial data, fusing and aggregating geospatial triples through intuitive, step-by-step processes, model-learning for automatic fusion recommendations and map-based UIs. Our focus is to build the FAGI framework based on real-world and diverse geospatial RDF datasets, so that we can capture the heterogeneity of data sources and provide an extended processing coverage to several use case scenarios. Figure 1 presents the architecture of the framework, the basic components of which are as follows:

- *Transformations component.* This is the component that normalizes the data w.r.t. the vocabularies used (i.e. RDF vocabularies for representation of geospatial features, OGC standards for the encoding of geometries in the form of literals, coordinate reference systems). Also, this component gathers and organizes quality metadata of the datasets, which either exist within the datasets, or are manually input by the user.

- *Geospatial processing component.* This component performs the whole process of geospatial indexing, to increase efficiency, calculation of topological relations, which are required to produce geometric similarity scores for fusion, and geospatial transformations of the geometric features of the entities, which are necessary for the actual fusion process, when a geometry or even a whole set of geometries need to be transformed to produce a final, fused geometry of higher quality.
- *Fusion component.* This component performs the core fusion functionality, including property mapping for both spatial and non-spatial properties, calculation of similarity scores for geospatial features of entities, and recommendation of fusion strategies. Furthermore, based on the calculated similarity scores, both on spatial and non-spatial properties of the entities, as well as on geospatial calculations on the neighborhoods of the interlinked entities, it suggests creation of new links between entities or rejection of existing links as false positives.
- *Learning component.* This component handles the training of machine learning models for automatic suggestion of fusion strategies for groups of triples, based on the individual characteristics of the interlinked geospatial entities, as well as the history of user actions w.r.t. selecting and validating fusion actions.

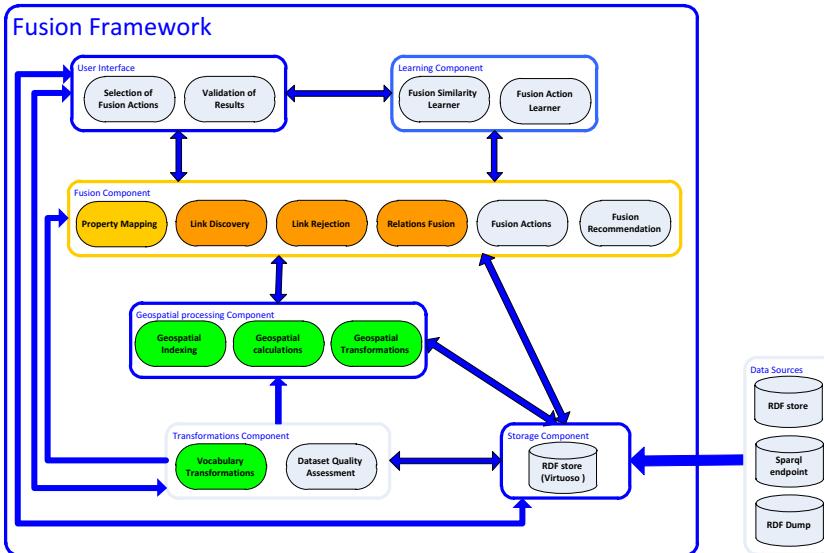


Fig. 1. Fusion framework architecture

Currently, most functionality regarding the Vocabulary Transformations and Geospatial Processing for fusion has been implemented, as depicted by the green-colored modules in Figure 1. This functionality is incorporated in two separate tools, FAGI-tr [7] (FAGI for transformations) and FAGI-gis [6] (FAGI for geospatial processing).

6 Conclusions

In this paper, we presented a framework, called FAGI, that handles the problem of fusing geospatial RDF data. Based on our findings w.r.t. shortcomings of existing methods

in the literature and to the specificities of the problem, we designed a modular framework that decomposes the several fusion subtasks into separate components/tools. Our future steps focus on enhancing the functionality and increasing the efficiency of the implemented tools, as well as implementing the learning module for automatic fusion recommendations.

Acknowledgements. This research is conducted as part of the EU project GeoKnow¹ FP7-ICT-2011-8, 318159.

References

1. Bleiholder, J., Naumann, F.: Declarative data fusion – syntax, semantics, and implementation. In: Eder, J., Haav, H.-M., Kalja, A., Penjam, J. (eds.) ADBIS 2005. LNCS, vol. 3631, pp. 58–73. Springer, Heidelberg (2005)
2. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. 41(1), 1–41 (2008)
3. Cobb, M.A., Chung, M.J., Miller, V., Foley III, H., Petry, F.E., Shaw, K.B.: A Rule-Based Approach for the Conflation of Attributed Vector Data. GeoInformatica 2(1), 7–35 (1998)
4. Chen, C.-C., Knoblock, C.A.: Conflation of Geospatial Data. In: Encyclopedia of GIS, pp. 133–140 (2008)
5. Chen, C.-C., Shahabi, C., Knoblock, C.A., Kolahdouzan, M.: Automatically and Efficiently Matching Road Networks with Spatial Attributes in Unknown Geometry Systems. In: Proc. of the 3rd STDBM (co-located with VLDB 2006) (2006)
6. FAGI-gis, <https://github.com/GeoKnow/FAGI-gis>
7. FAGI-tr, <https://github.com/GeoKnow/FAGI-tr>
8. Knap, T., Michelfeit, J., Necasky, M.: Linked Open Data Aggregation: Conflict Resolution and Aggregate Quality. In: Proc. of the 2012 IEEE 36th Annual Computer Software and Applications Conference, pp. 106–111 (2012)
9. Michelfeit, J., Knap, T.: Linked Data Fusion in ODCleanStore. In: Proc. of the International Semantic Web Conference (Posters & Demos) (2012)
10. Mendes, P.N., Muhleisen, H., Bizer, C.: Sieve: linked data quality assessment and fusion. In: Proc. of the 2012 Joint EDBT/ICDT Workshops, pp. 116–123 (2012)
11. Ngonga Ngomo, A.-C.: Learning conformation rules for linked data integration. In: Proc. of the 7th International Workshop on Ontology Matching (2012)
12. Nikolov, A., Uren, V.S., Motta, E., De Roeck, A.: Integration of Semantically Annotated Data by the KnoFuss Architecture. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 265–274. Springer, Heidelberg (2008)
13. Open Geospatial Consortium Inc. OGC GeoSPARQL standard - A geographic query language for RDF data,
https://portal.opengeospatial.org/files/?artifact_id=47664
14. Preparata, F., Shamos, M.I.: Computational Geometry: An Introduction. Springer, New York (1985)
15. Stankutė, S., Asche, H.: An Integrative Approach to Geospatial Data Fusion. In: Gervasi, O., Taniar, D., Murgante, B., Laganà, A., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2009, Part I. LNCS, vol. 5592, pp. 490–504. Springer, Heidelberg (2009)
16. Stankutė, S., Asche, H.: Improvement of spatial data quality using the data conflation. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011, Part I. LNCS, vol. 6782, pp. 492–500. Springer, Heidelberg (2011)

¹ <http://geoknow.eu/>

17. Saalfeld, A.: Conflation: Automated Map Compilation. *Int. J. Geogr. Inf. Sci.* 2(3), 217–228 (1988)
18. Schultz, A., Matteini, A., Isele, R., Mendes, P., Bizer, C., Becker, C.: LDIF-A Framework for Large-Scale Linked Data Integration. In: Proc. of the 21st International World Wide Web Conference, Developer Track (2012)
19. Ware, J.M., Jones, C.B.: Matching and Aligning Features in Overlayed Coverages. In: Proc. of the 6th ACM Symposium on Geographic Information Systems (1998)
20. Walter, V., Fritsch, D.: Matching Spatial Data Sets: a Statistical Approach. *Int. J. Geogr. Inf. Sci.* 5(1), 445–473 (1999)

Online Reasoning for Ontology-Based Error Detection in Text

Fernando Gutierrez¹, Dejing Dou¹, Stephen Fickas¹, and Gina Griffiths²

¹ Computer and Information Science Department

University of Oregon

Eugene, Oregon 97403, USA

{fernando,dou,fickas}@cs.uoregon.edu

² Special Education and Clinical Sciences Department

University of Oregon

Eugene, Oregon 97403, USA

ginag@uoregon.edu

Abstract. Detecting error in text is a difficult task. Current methods use a domain ontology to identify elements in the text that contradicts domain knowledge. Yet, these methods require manually defining the type of errors that are expected to be found in the text before applying them. In this paper we propose a new approach that uses logic reasoning to detect errors in a statement from text online. Such approach applies Information Extraction to transform text into a set of logic clauses. The logic clauses are incorporated into the domain ontology to determine if it contradicts the ontology or not. If the statement contradicts the domain ontology, then the statement is incorrect with respect to the domain knowledge. We have evaluated our proposed method by applying it to a set of written summaries from the domain of Ecosystems. We have found that this approach, although depending on the quality of the Information Extraction output, can identify a significant amount of errors. We have also found that modeling elements of the ontology (i.e., property domain and range) likewise affect the capability of detecting errors.

Keywords: Information Extraction, Ontology, Consistency Checking.

1 Introduction

As Information Extraction, the task of automatically identifying entities and relationships in text, moves from the analysis of scientific documents to Internet textual content, we cannot rely completely on the assumption that the content of the text is correct. Indeed, in contrast to scientific documents, which are peer reviewed, Internet content is not verified for the quality and correctness. So, it seems reasonable to consider, as part of the process of extracting textual information, mechanisms and techniques that allow to differentiate between correct and incorrect information.

However, it is not easy to determine the correctness of a sentence; hence, it has been addressed only indirectly. Research from the educational oriented field

such as automatic text grading has mainly treated incorrectness as low similarity to a gold standard. Automatic grading systems based on Latent Semantic Analysis [13] and *n-gram* co-occurrence [14] try to determine how similar a student's summary or essay is with respect to a *perfect* summary or essay. If the student's writings has low similarity to the gold standard, it means that it is *less correct*. However, low similarity can still be obtained in the process of a correct text, such as if the text was written in an unexpected fashion, or if the text content is broader than the gold standard. On the other hand, incorrectness can be identified in the presence of contradiction. The research area of Contradiction Detection [20], an extension of Textual Entailment, intends to identify in text pair of sentences that cannot be true at the same time (i.e., logic contradiction). By identifying specific lexical and syntactical elements, the semantics of the sentences are captured and compared. However, since Contradiction Detection is limited to the content of the text itself in order to support the correctness of the contradicting sentences, it cannot determine with certainty which sentence of the pair is false.

Following a different approach, we have proposed an ontology-based error detection method using pre-defined errors for extraction rules [5] and machine learning based patterns [6]. An ontology provides a formal knowledge representation of a domain through concepts, relationships, and axiomatic constraints. By including a domain ontology into our approach, we have a formal model that allows the use of logic reasoning services, such as contradiction detection, plus a set of correct facts from the domain (e.g., relationships between concepts of the domain). So, if a sentence contradicts the domain ontology, then the sentence can be considered to be incorrect with respect to the domain knowledge. Our approach incorporates a heuristic algorithm to generate domain-inconsistent facts that are encoded into an Ontology-based Information Extraction system. Although this approach can identify incorrect sentences based on the domain, it is limited by the set of expected errors defined in our heuristic algorithm. Such heuristic algorithm has a set of manually defined rules that generate axioms by violating constraints in the domain ontology. For example, if the domain ontology defines that concept *A* and *B* cannot share elements (i.e., they are disjoint), then the heuristic algorithm will generate an axiom of elements that result from the union of concepts *A* and *B*. This inconsistent axioms is encoded into an extraction pattern that identifies incorrect sentences that state that elements of concept *A* are also elements of concept *B*. In this way our previous method was able to identify errors (i.e., incorrect statements). However, this is also its main limitation. Our previous approach could only recognize incorrect sentences if they were part of the training set or very similar to a sentence in the training set. New sentences could not be judged correctly.

In this paper, we propose a new method to detect incorrectness by online inference. This method considers incorrectness as inconsistency with respect to the domain, but instead of defining a set of expected error rules, we incorporate a reasoner to determine if a sentence is inconsistent with the domain ontology. To this end, we apply a variation of an Ontology-based Information Extraction

(OBIE) process. OBIE is a subfield of Information Extraction that incorporates a domain ontology to guide the extraction process. In our approach, instead of having the ontology guiding the extraction process, the information extraction is performed based on structural elements from the text, while the ontology is used to validate the correctness of the extracted statements. Although this approach differs from the definition of OBIE [24], we argue that it is still an OBIE process since the approach relies on the domain ontology to determine the correctness of each statement. The main goal of this approach is to provide the most complete analysis of the correctness of a text. In that sense, the extraction process intends to extract every possible relationship in the text, while the inconsistency analysis use the complete ontology to infer the correctness or incorrectness of the extracted relations. In the evaluation of our proposed method, we found that our method can identify accurately the classification of a significant amount of sentences. We also found that the quality of the IE process affects the overall performance of our proposed method.

The rest of the paper is organized as follows. We provide a brief review of some related work in Section 2. Then we demonstrate our ontology-based error detection method in Section 3. We report our experimental results in Section 4, and discuss some observations from our case study in Section 5. We conclude the paper by summarizing our contributions and future goals in Section 6.

2 Background

In this work, we propose a method to identify incorrectness in text by combining logic reasoning and domain knowledge. This has led us to consider research regarding Information Extraction, Consistency Checking and Ontology Debugging to provide us with ideas that can help us to reach our goal.

2.1 Information Extraction

Information Extraction (IE) is the task of automatically acquiring knowledge by transforming natural language text into structured information, such as a knowledge base [24]. In the process of extracting, IE attempts to retrieve specific elements from text such as concepts, relationships, or instances. For example, from the sentence “Albert Einstein was born in Ulm,” an IE process will identify *Albert Einstein* and *Ulm* as relevant instances that are connected by the relationship *born_in*. This leads to the extraction of the relationship *born_in(Albert Einstein, Ulm)*. However, this transformation of information is in general not a trivial process because of the inherent ambiguity of natural language text. A fact can be stated in many ways, and a statement can have more than one interpretation. The complexity of extracting information from text has kept IE from being more widely adopted, with most IE systems being implemented only for specific domains.

The complexity of extracting information from text can be mitigated by the inclusion of domain knowledge in the process of capturing the semantic elements

from the text. The subfield of Ontology-based Information Extraction (OBIE) uses a formal representation of the domain (i.e., ontology) to guide the extraction process [24]. The guide offered by the domain ontology allows OBIE to focus on the extraction of specific concepts and relationships of the domain.

By considering the amount of human intervention in the preparation required for the system's deployment, three different strategies have been proposed to accomplish this transformation: *supervised*, *semi-supervised*, and *unsupervised*. A *supervised* IE systems require significant amount of intervention before being utilized. This intervention comes either in the form of labeled data in the case of IE system that are based on machine learning, or handcrafted patterns in the case of extraction rules [5]. Since the extraction process is tuned to focus on specific entities and relationships, most OBIE systems follow a supervised strategy. Ontological elements are included in the creation of extraction patterns and in the labeling of data. Because of this close labeling process, the uncertainty about the extracted relationship and what it represents is small. The close labeling process also allows supervised systems to extract implicit relationships, which in most cases cannot be identified by semi-supervised or unsupervised systems.

In the case of *semi-supervised* IE systems, the extraction process uses the connection between sentences and hand built knowledge bases to learn information extraction models and patterns for specific relationships. In contrast with supervised systems which have an explicit link between text and instances, semi-supervised systems have to discover this connection. In some cases the connection is clear, for example *Kylin* system exploits the relation between Wikipedia's *infoboxes* and articles [25]. On other cases, where the corpus of text is the Web [17], or the text is in a different language than that of the knowledge base [2], the connection is not that evident.

Semi-supervised system will consider a known relationship between a pair of entities (e.g., *place_of_birth(Immanuel Kant, Königsberg)*) or a known instance of a concept (e.g., *country(United States)*). This known tuple can be obtained from a knowledge based (e.g., DBpedia) or similar (e.g., Wikipedia's infoboxes). The system then will search in the text for sentences containing the entities from the relationship, such as "Immanuel Kant was born in 1724 in Königsberg, Prussia, as the fourth of nine children (four of them reached adulthood)." The selected sentences will then be transformed into vectors and enriched with grammatical information (e.g., *part-of-speech*). The transformed sentences will be used as training data for machine learning methods, such as Conditional Random Fields [25]. The learned machine learning models are then used to extract new instances and relationships.

Finally, *unsupervised* IE system can perform extractions without requiring labeled data or specific pattern construction. They used general patterns based on lexical and syntactical features from the text to identify relationship between entities. Initially these patterns could only identify *hyponymy* relationships [9]. However, continuous extensions to these patterns by the inclusion of more

complex features (e.g., syntactic dependencies [15]) has led to unsupervised IE systems to extract a wide variety of relationships [3].

One of the unsupervised extraction patterns proposed by Hearst [9] is “ $NP_0, NP_i^{i=1\dots n}$ (*and/or*) *other* NP ,” which allows the extraction of the relation $[hyponymy(NP, NP_i)]^{i=0\dots n}$. For example, when the previous pattern is applied to the sentence “France, Italy, and other European countries...” we obtain the relationships $hyponymy(France, European_country)$ and $hyponymy(Italy, European_country)$.

2.2 Consistency Checking

Through concepts, relationships, and constraints, an ontology provides a vocabulary and a model of the domain it represents. In this work, we consider Description Logic based ontologies, as those described through the Web Ontology Language (OWL) [1]. OWL is the standard ontology language proposed by the World Wide Web Consortium (W3C).

Description Logic (DL) is a fragment of first-order logic that is decidable, and it has sound and complete reasoners, such as HermitT [18]. DL employs Boolean constructors, such as conjunction (\sqcap), disjunction (\sqcup), existential (\exists) and value (\forall) restrictions, to describe concepts and roles (i.e., relationships). DL consists of a *TBox* \mathcal{T} , which is a set of *general concept inclusions* (GCI) of the form $C \sqsubseteq D$, for concepts C and D . It also has a set \mathcal{A} , called *ABox*, of concept and role assertions of the form $C(a)$ and $R(a, b)$, for concept C , role R and individuals a and b . In some DL languages, there is an *RBox* \mathcal{R} that consists of complex role constructions such as role inclusion ($R_1 \sqsubseteq R_2$), role disjointness, reflexivity, and others. A knowledge base \mathcal{K} in DL is conformed as the tuple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$. In the case of DL languages that do not have *RBox*, such as \mathcal{ALC} , the tuple takes the form $(\emptyset, \mathcal{T}, \mathcal{A})$ or $(\mathcal{T}, \mathcal{A})$.

The semantics of a DL knowledge base \mathcal{K} is defined by an interpretation $I = (\Delta^I, \cdot^I)$. The interpretation consists of a domain (Δ^I) and a mapping function (\cdot^I). The function maps the concepts, roles and assertions of the knowledge base to the domain. If I satisfies all axioms of \mathcal{K} , then I is a model of \mathcal{K} , which makes \mathcal{K} consistent. A basic service of a DL reasoner is to determine if a knowledge base \mathcal{K} is satisfiable. Determining satisfiability is fundamental since other types of inferences can be reduced to unsatisfiability, such as entailment [11]. A DL reasoner demonstrate the satisfiability of a knowledge base by constructing an abstract model through a tableau algorithm. The algorithm creates a sequence of *ABoxes*, where each new *ABox* (\mathcal{A}_i) is the result of the application of derivation rules to a previous *ABox* (\mathcal{A}_{i-1}) [18]. Following are commonly used tableau derivation rules for DL:

- Given $C \sqsubseteq D$ and an individual s , derive $(\neg C \sqcup D)(s)$.
- Given $(C \sqcup D)(s)$, derive $C(s)$ or $D(s)$
- Given $(C \sqcap D)(s)$, derive $C(s)$ and $D(s)$.
- Given $(\exists R.C)(s)$, derive $R(s, t)$ and $C(t)$ for a new individual t .
- Given $(\forall R.C)(s)$ and $R(s, t)$, derive $C(t)$.

Through the derivation rules, the consistency checking algorithm can terminate by reaching one of two possible cases. In the first case, if the derivation leads to a contradiction, the knowledge base \mathcal{K} is determined to be inconsistent, and the algorithms terminates. The second case, if no derivation rule can be applied, then the algorithm terminates, and the derived *ABox* (\mathcal{A}_i) represents a model for the knowledge base \mathcal{K} (i.e., \mathcal{K} is consistent).

2.3 Ontology Debugging

Since we consider incorrectness as an inconsistency with respect to the domain ontology, it seems reasonable to consider research regarding ontology inconsistency. However, DL reasoners cannot determine the origin of the inconsistency. There are three main approaches when dealing with inconsistency [8]: preventing ontology inconsistency [7], fixing ontology inconsistency [10,21], or reasoning with inconsistency [12]. We believe that research in fixing ontology inconsistency can provide some insight in incorrectness detection in text because both tasks focus on determining logical contradiction and its origin.

The process of fixing an inconsistent ontology is known as *Ontology Debugging* [4]. The task of Ontology Debugging has been addressed using properties of the underlying description logic (DL) language. These methods try to identify the minimal inconsistent sub-ontology, which is the set of axioms that make the ontology inconsistent. Horridge et al. [10] have identified two main stages in the process of determining the origin of inconsistency in an ontology. In the first stage they determine the set of all inconsistent sub ontologies (i.e., *inconsistency justification*), while in the second stage they construct the *minimal* inconsistent sub ontology from the previous set (i.e., *ontology repair*).

In order to find the set of all the inconsistent sub ontologies, Schlobach and Cornet [21] proposed to determine the *minimal unsatisfiability-preserving sub-TBoxes* (MUPS) of unsatisfiable concepts of the ontology. The *MUPS* of a concept is the set of axioms that cause the concept to be unsatisfiable. In their original work, Schlobach and Cornet [21] obtained the MUPS of each concept through a modified \mathcal{ALC} reasoner that inserts traceable labels in the axioms when performing consistency checking. But because this approach does not scale well to more expressive DL languages, Schlobach et al. [22] offer an alternative mechanism to identify each concept MUPS. Based on Huang et al. selection function [12] for reasoning with inconsistent ontologies, Schlobach et al. used an informed search to find concept-relevant axioms. The set produced by the selection function is then pruned by removing axioms that do not affect the unsatisfiability of the concept MUPS.

On the other hand, in Horridge et al.'s approach [10], the inconsistent subsets of the ontology are obtained by dividing the ontology and testing the consistency of the partition. The intuition is that the cause of inconsistency will be in the inconsistent part of the ontology, and not in the consistent part. It is important to note that it is possible to remove accidentally the inconsistency when dividing the ontology. To avoid missing an inconsistent subset, Horridge et al. approach also analyzes the recombination of the divided parts.

Once the set of all inconsistent sub ontologies are obtained, the minimal inconsistent sub ontology can be identified. In the case of Schlobach and Cornet's approach, from the *MUPS* the *minimal incoherence-preserving sub-TBoxes* (MIPS) are determined, which are unsatisfiable sub-TBoxes that can become satisfiable if one atomic concept is removed. Because each element of the MIPS set comes from some MUPS, the MIPS set is contained in the union of all MUPS of the original *TBox*. Although the *MIPS* already identifies the minimal inconsistent set of the ontology, Schlobach and Cornet's approach offers an even more fine grained solution. Because inconsistencies can be interpreted as the effect of overspecified concepts [21], it is possible to identify the actual concepts that are clashing by generalizing the axioms of the MIPS. This new set is obtained by the *generalized incoherence-preserving terminology* (GIT), where all elements in an axiom of the MIPS which do not affect its unsatisfiability are removed.

On the other hand, Horridge et al. used the Hitting Set Tree algorithm [19] to identify the minimal inconsistent sub ontology from the set of inconsistent sub ontologies. Reiter proposed the Hitting Set Tree (HST) [19] as a form to determine the diagnosis a faulty system. In a faulty system there can be multiple reasons that explain the actual error (i.e., *conflict sets*). Yet in order to correct or fix the system, it is necessary to identify the minimal conflict set (i.e., diagnosis). Reiter's HST finds the diagnosis by learning how the conflict sets intersect. The HST algorithm iteratively searches or accesses the set of conflict sets. It constructs a tree where each node indicates a conflict set and the edges indicate an element of the conflict set. The set formed by the labels on the edges along a branch of the HST corresponds to one diagnosis. In the case of ontology inconsistency, the HST identifies the minimal inconsistent sub ontology by constructing a tree that has as nodes the inconsistent sub ontologies.

These methods can help in the task of incorrectness detection in text by providing mechanisms to determine the reason a sentence is incorrect with respect to the domain ontology.

3 Methodology

In the present work we propose online incorrectness inference, a method to detect logic errors of text content by incorporating logic reasoning and domain knowledge. The error detection comes from determining if the text is logically consistent with respect to the modeled domain knowledge (i.e., ontology). The consistency of the text can be checked by adding sentences, as extracted entities and relationships, to the domain ontology and verifying its consistency.

The online incorrectness inference approach consists of three steps. In the first step, statements are extracted from the text through an Information Extraction process. The extraction process provides a transformation of the statement from its original textual form into a logical bound form. As a second step, the extracted statements are formalized and added to the domain ontology. The formalization intends to transform the extracted tuples into a form compatible with the ontology. The third and final step is to determine the correctness of the statement

through a logic-based service provided by an ontology: consistency checking. If the domain ontology becomes inconsistent after the extracted sentence is added into it, then the sentence is incorrect with respect to the domain.

In the following sections we provide details of each one of the steps of our proposed method.

3.1 Information Extraction

In order to evaluate the correctness of a sentence, we first transform unstructured text into structured information. As previously mentioned, there are three main strategies to IE depending on the level of human intervention (i.e., data preparation). However, because our approach intends to determine the correctness of each statement presented in the text, not all three strategies are suited for our approach. Supervised IE cannot provide a complete extraction from the text since the process is guided by known labeled data and predefined patterns. Similarly, semi-supervised IE systems are guided to extract relationships based on sets of known individuals. Plus, in order to provide quality extraction, semi-supervised IE requires a significant set of training individuals.

For the present work, we have chosen the unsupervised strategy followed by the Open Information Extraction system *OLLIE* [15]. Open Information Extraction systems intend to extract binary relationships without using any training data (or handcraft patterns). The main goal behind this approach is to offer an IE system that can scale to the Web. To do this, Open Information Extraction follows a set of general patterns to extract every possible relationship from a text [3]. In the case of *OLLIE*, the patterns are built by generalizing extractions with high confidence (i.e., high quality extraction). The set of high quality extractions is obtained from Open Information Extraction system *ReVerb* [3], which uses a verb-based patterns to identify relations in text. These extractions (e.g., tuples) have two constraints: they contain solely proper nouns as entities participating in the extracted relation, and they have a high confidence value. Then, similar to semi-supervised IE systems, *OLLIE* gathers a set of sentences that contain the entities and relations from the extracted tuples. To avoid collecting sentences that might introduce errors, *OLLIE* only gathers sentences with a structure that is centered in the elements of the extracted tuple, i.e., elements of the relation must be in a linear path of at most size four in the dependency parse [15]. From the selected sentences, *OLLIE* learns a set of general extraction patterns. If the structure of a sentence meets a set of requirements (e.g., the relation is in between the two entities in the sentence), a pure syntactic pattern can be learned from the sentence (e.g., most general pattern). If the structure of the sentence does not meet the requirements, lexical aspects of the sentence are considered in order to produce a general pattern. These generalized patterns are used to extract new tuples from text.

Because we focus on determining the correctness of the text's content, we will consider *OLLIE* as a blackbox component of our system. *OLLIE* will produce a set of binary relationships from the text, which will be passed to the reasoner.

For example, from the sentence “Scavengers feed from dead organisms,” OLLIE will produce the tuple *feed(Scavengers, dead organism)*.

3.2 Transformation of Extracted Relationships

Before adding the extracted relationships to the ontology to determine their correctness, we need to solve first a lexical gap that might exist between the text and the ontology. Although the text and the ontology belong to the same domain, it is very possible that the selection of words to represent concepts and relationships might differ. So, to be able to use the domain ontology to evaluate the correctness of the text’s semantics, we will need a mapping mechanism that can allow us pass from the vocabulary of the extracted entities and relationships to the vocabulary of the ontology.

In the case of relationships at the terminology level of the domain (i.e., *TBox*), the mapping can be accomplished with simple mechanisms, such as gazetteers (i.e., list of words) of ontological terms, or through *WordNet* to find synonyms of terms [16]. However, in the case of extracted relationships from the assertion level of the domain (i.e., *ABox*), the solutions might not be so trivial. Most domain ontologies have very few individuals, if any at all. Because the domain ontology provides a general representation of the domain, it is very unlikely that it contains a large number of specific examples. We believe that the case of managing extracted relationships from the assertion level will require more sophisticated techniques, such as reasoning in the presence of uncertainty.

In the present work, because we are dealing with the writings on the generalization of domain knowledge (i.e., summaries), it seems reasonable to consider gazetteers for mapping vocabularies. We have defined two gazetteers: one for managing concepts, and another for managing relationships. In the case of the gazetteer for managing concepts, an extracted entity will lead to the equivalent ontological concept. For example, both *dead organisms* and *dead animals* lead to the concept *Dead Organism*. In the case of managing relationships, because a relationship might have different meaning depending on other elements in the sentence, we consider both subject entity and relation to determine the ontological property. For example, the concept *Carnivore* and the relation *feed* will lead to the property *feed_from_herbivore*, while concept *Herbivore* and relation *feed* will lead to the property *feed_from_producer*. Both gazetteers are generated by considering a subset of extracted relationships from the data set.

3.3 Determining Correctness of a Sentence

Once we have extracted all the relations from the text (e.g., “Autotrophs produce their food,” to *produce(Autotrophs, food)*), and the relations have been mapped to the vocabulary of the ontology (e.g., *produce(Autotrophs, food)* to *Autotrophs ⊑ ∃produce.Food*), we proceed to analyze the correctness of the statements by using consistency checking. We propose the use of a logic reasoner to determine the correctness of text.

We have identified two approaches when analyzing text extractions: single sentence analysis and multiple sentence analysis. In single sentence analysis, we intend to determine the correctness of text by considering one sentence at a time. Under this approach the semantic content of each sentence is considered independent from the rest of the text. In the case of multiple sentence analysis, a group of sentences from the text are analyzed as set of clauses. Although the analysis of multiple sentences leads to a higher computational complexity, it allows us analyze the correctness between sentences. There are cases where sentences can be consistent when considered independently, but become inconsistent when analyzed as a set.

In this work, we focus on perform single sentence analysis. Each sentence will be included into the domain ontology independently. After the analysis of the sentence has concluded, the sentence's relationship will be removed from the domain ontology. Multiple sentence analysis will be discussed in Section 6.

Type of Sentence. In the previous work, we identify three type of sentences [5]: correct, incorrect, and incomplete. In the present work, we offer a more precise definition of them.

A sentence is considered to be *correct* if it can be entailed from the domain. We argue that consistency is not a sufficient requirement to define correctness since a sentence could be undefined in the domain but still being consistent with it. On the other hand, if a sentence is entailed by the domain, then there is a subset of the domain from which we can derive the semantic content of the sentence. For example, given the domain ontology \mathcal{O} and the axioms $a_1 : Producer \equiv Autotroph$ and $a_2 : Producer \sqsubseteq \exists produce.Food$, with $a_1, a_2 \in \mathcal{O}$, the sentence “Autotrophs produce their food” (i.e., $Autotroph \sqsubseteq \exists produce.Food$) is correct because $\mathcal{O} \models (Autotroph \sqsubseteq \exists produce.Food)$.

A sentence is considered to be *incorrect* if it is inconsistent with respect to the domain ontology. In other words, a sentence is incorrect if it violates a domain constraint. In order to determine the consistency of a sentence, we must add it to the domain ontology before doing consistency checking. For example, given the domain ontology \mathcal{O} and the axioms $a_1 : Producer \sqsubseteq \neg Carnivores$ and $a_2 : Producer \sqsubseteq \exists produce.Food$, with $a_1, a_2 \in \mathcal{O}$, the sentence “Carnivores produce their food” (i.e., $Carnivores \sqsubseteq \exists produce.Food$) is incorrect because $\mathcal{O} \cup Carnivores \sqsubseteq \exists produce.Food \models \perp$.

Finally, if a sentence is consistent with the domain ontology, but it cannot be entailed, then it is considered an *incomplete* sentence. By incomplete, we mean that it is not possible to determine the sentence is correct or not. Under the open world assumption, we do not know the truth value of entities and relationships that are not defined in the ontology. Let us consider the case where the axioms $a_1 : Tree \sqsubseteq Producer$ and $a_2 : Producer \sqsubseteq \exists produce.Food$, with $a_1 \notin \mathcal{O}$ and $a_2 \in \mathcal{O}$, the sentence “Trees produce their food” (i.e., $Tree \sqsubseteq \exists produce.Food$) is incomplete because $\mathcal{O} \cup (Tree \sqsubseteq \exists produce.Food) \not\models \perp$ and $\mathcal{O} \not\models (Tree \sqsubseteq \exists produce.Food)$.

However, to be able to determine if a sentence is consistent or entailed by the domain, the domain ontology needs to meet two requirements: consistency and completeness. The first requirement, consistency of the domain ontology, is fundamental for determining the correctness of a sentence. As stated by Haase and Völker [8], an inconsistent ontology is meaningless since anything can be inferred from a set of contradicting axioms. This means that all sentences can be entailed by an inconsistent domain ontology, making all of them correct. In the case of the second requirement, completeness of the domain ontology, it has more practical implication. If the domain ontology does not have all the axioms (e.g., concepts, relationships) required to analyze the text, it is most likely that a sentence will be labeled as incomplete rather than correct or incorrect.

In this work we have selected Hermit as the reasoner because of its higher efficiency (i.e., hypetableau reasoning algorithm) [18].

Explanation of Incorrectness. We expect that in the case of incorrect sentence, our method should provide an explanation of why the sentence contradicts the domain ontology. For that purpose, we have considered the ontology debugging solution proposed by Horridge et al. [10]. As previously mentioned, Horridge et al. *explanation* approach integrates Reiter's Hitting Set Tree (HST) [19] to identify the minimal inconsistent sub-ontology, i.e., subset of axioms from the ontology that cause the inconsistency.

Horridge et al.'s approach first determines a set of inconsistent sub ontologies (e.g., $\mathcal{O}_1 \dots \mathcal{O}_n \subseteq \mathcal{O}$ with $\mathcal{O}_1 \models \perp \wedge \dots \wedge \mathcal{O}_n \models \perp$). These sub ontologies are obtained by dividing the ontology and checking their consistency. The intuition behind this step is that the cause of inconsistency of the ontology will be located in a section of the ontology (i.e., sub ontology), and not across the whole ontology. From the set of sub ontologies, the approach constructs a Hitting Set Tree (HST) [19]. The Hitting Set Tree algorithm provides a mechanism to identify a minimal set of elements that can provide *minimal coverage* of over a set of clauses. In the case of inconsistent ontology, HST identifies the minimal set of axioms that participate in all the inconsistent sub ontologies. Reiter has offered a set of optimization and pruning mechanisms to reduce the computational cost of HST [19]; yet, since HST requires multiple consistency checks, it is a very costly method.

Horridge et al.'s approach has been incorporated into popular DL reasoners, such as Hermit [18].

4 Experiments

4.1 Data Set

In this work we will use a set of summaries collected on an earlier study by Sohlberg et al. [23] that looked at the use of electronic strategies (eStrategies) for reading comprehension for college students. As part of the study, students were asked to provide oral summaries of each article they had read, where each

article is roughly 3 pages in length (4 articles were used). The oral summaries were manually transcribed into text form. From the Sohlberg et al.'s collection, we will consider for the present work 18 summaries from the Ecosystems article. The summaries vary in length from a pair of sentences to 60 sentences. A section of a summary from the Ecosystem set can be seen in Example 1.

*In the ecosystem there are different types of animals.
Producers make their own food from the environment.
Consumers eat other consumers and producers.
The producers are plants, trees, algaees.
...*

Example 1: Part of a summary from the Ecosystems set.

Because the summaries are originally *oral summaries*, they slightly differ from written ones (as it can be seen in Example 1). The transcribed summaries contain repeated statements, and in some cases there are contradictions when considering the summary as a whole. However, because we focus on resolving the correctness of the text content one sentence at a time, these cohesion issues do not affect our analysis.

The summaries have been preprocessed in order to simplify the extraction process. The preprocessing has been focused on resolving anaphoras and cataphoras (e.g., pronouns) and on correcting misspellings. The summaries have also been labeled at the sentence level according to the correctness of their content. The labeled summaries have been used as the gold standard for the purpose of evaluation.

4.2 Ontology

In this work, we constructed an ontology for the domain of Ecosystems. We used the introductory article that the student used for their summaries. The construction of the ontology is constrained to explicit facts from the domain knowledge defined by the article, and does not include facts from the entire domain of Ecosystems. By keeping our ontology centered on the introductory article, we intend that the ontology will better cover concepts and relationships from the students summaries, which are also solely based on the article.

Table 1. Statistical information about the ontology

Element type	Number of element
Concepts	55
Relationships	30
Axioms	224

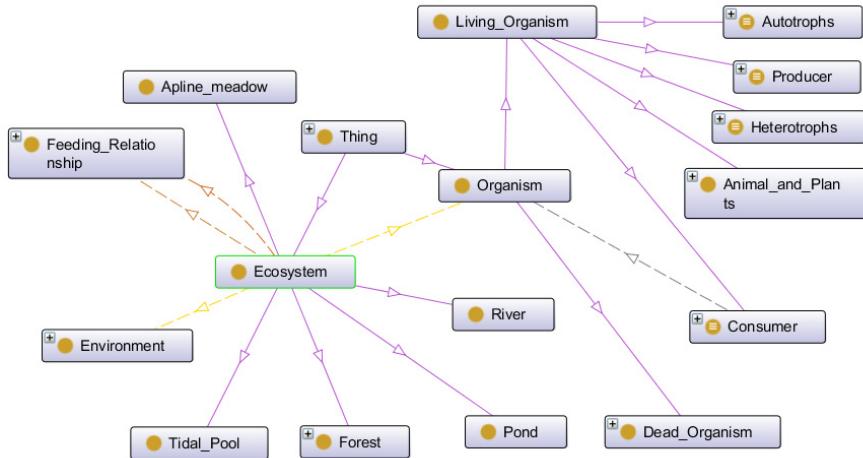


Fig. 1. Graphical representation of a section of the Ecosystems ontology

Because of the strict construction criteria, the ontology has many concepts that do not have a membership relationship with another concept, as well as not having instances. This is originated by the nature of the Ecosystems article. Because the article is an introduction to the domain, a broad set of concepts and relationships of the topic are presented rather than details, such as specific examples. In Figure 1 we presents a graphical representation of a part of the Ecosystems ontology.

It must be mentioned that, although the ontology used in our present approach is similar to the one used in our previous work [5], there is a significant difference in the number of axioms of each ontology. In order to determine incorrectness based on logic contradiction, the ontology for the present work incorporates a large set of constraints, such as disjointness between classes, and strictly defines domain and range for each property.

4.3 Comparison Method

To obtain a better understanding of how well our proposed method performs, we have considered two comparison methods. The first method is our previous work [5], which is, to the best of our knowledge, the only method for error detection in text based on ontologies. As previously mentioned, our previous approach manually defines domain inconsistent axioms by violating ontological constraints. These inconsistent axioms are encoded into extraction patterns that can detect incorrect sentences before the extraction process begins (i.e., precomputed approach).

For comparison, we have used the same set of rules manually defined before. We created the extraction rules by using the domain ontology and considering the content documents. Because it is possible to generate a large set of inconsistent axioms from the domain ontology, we use the content of the four documents to limit the number of extraction rules that need to be generated. This led to 31 extraction rules to identify correct sentences, 16 extraction rules to identify incorrect sentences, and five extraction rules to identify incomplete sentences.

The second comparison method is a variation to our proposed method that replace the IE process with *manual extraction*. This variation can provide us with insight of how the mapping and reasoning steps perform when analyzing correctness. Because currently available IE implementations are not 100% accurate, the overall performance of error detection might be affected by the IE process. The use of *manual extractions* can lead to an overall performance depending on directly the performance of the mapping and reasoning steps of our approach. We have constructed a data set formed by binary relationships manually extracted from the 18 summaries. These manually extracted relationships are then analyzed by our approach to determine their correctness.

For the mapping step, we use the same gazetteers for both proposed approach (i.e., automatic extraction) and the *manual extraction* method. The gazetteers were constructed by observing extracted relationships from 40 sentences taken from four of the 18 summaries.

5 Results and Discussion

We use the traditional IE metrics (precision, recall, F1 measure) to measure the quality of the performance of our proposed new method and compare it with our previous method using precomputed errors. Precision measures the correctness of the labeling process, i.e., what percentage of the sentences that are labeled correct are actually correct. Precision is calculated by dividing the number of correct extractions or *true positives* (*tp*) by the total number of extractions, which are *true positives* plus *false positives* (*tp + fp*).

$$P = \frac{tp}{tp + fp}$$

Recall measures the completeness of the labeling process, i.e., what percentage of all correct sentences are labeled as such. Recall is calculated by dividing the number of correct extractions (*tp*) by the total number of instances that should be extracted, which are *true positives* plus *false negative* (*tp + fn*).

$$R = \frac{tp}{tp + fn}$$

Finally, F1 is the simplest and most commonly used form of F-measure in OBIE, and it is the harmonic mean between precision and recall.

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

From Table 2, we can say that in the case of automatic extraction approach it is possible to determine with high precision the correctness of a sentence with respect to the domain by logic reasoning. However, there is a significant amount of sentences that, although contained in the domain, are considered to be unrelated to it. There are a significant amount of cases where the IE process extracted phrases as entities. Although this is not strictly incorrect, most of these phrases represented something more than only a domain concept. This leads to a lower recall. On the other hand, although not all correct and incorrect sentences were captured, the sentences that were labeled as correct are all semantically correct sentences. The same goes with the incorrect sentences.

The perfect precision (i.e., 100%) obtained by both automatic and manual extraction approaches in the case of correct and incorrect sentences might seem unrealistic. However, it is the natural outcome given the underlying method used in the process (i.e., reasoning). If one sentence was labeled as correct when it was actually incorrect, it would mean that reasoning process used to determine the label the sentence is not accurate. However, as previously mentioned, we are using a DL reasoner (i.e., HermiT) which is sound and complete. So, once the semantic elements of a sentence are mapped to the ontology, the reasoner can accurately determine if it contradicts the domain ontology or not.

In the case of *manually extracted* relations, we can observe an increment in the recall with respect to the automatic extraction approach, with the same level of precision. This result indicates that the quality of the extraction process has a significant effect in the detection of correctness, it is not the only factor affecting the recall of correct and incorrect sentences. In the case of *manual extractions*, the error in determining the correctness of a sentence can be explained by the mapping between extractions and ontology. The correct (and incorrect) sentences that were labeled as incomplete are cases where the mapping procedure failed to connect extraction entities with ontological concepts.

Table 2. Precision, recall, and F1 measure for the proposed method (automatic and manually extraction) and for the *precomputed* approach [5]

Sentence	Automatic Extraction			Manual Extraction			Precomputed approach		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Correct	100%	40.9%	58.1%	100%	80.23%	89.0%	91.9%	83.3%	87.4%
Incorrect	100%	41.3%	58.4%	100%	88.63%	93.97%	97.4%	88.6%	92.8%
Incomplete	89.5% %	100%	94.4%	74.71%	100%	85.52%	66.7%	80.0%	72.7%

When compared with our previous approach, precomputed error detection, both our proposed automatic extraction and manual extraction methods are more accurate when identifying incorrect sentences. On the other hand, because our previous approach seeks specific pre-defined patterns in the text, it has a higher recall. However, the *precomputed error* has higher deployment conditions (i.e., overhead) since the extraction rules need to be created by domain and ontology experts.

6 Conclusions and Future Work

In this work, we propose a new approach to detect errors in text. By applying logic reasoning over a domain ontology, we can determine the correctness (and incorrectness) of text content. Our method applies Information Extraction to transform text into a set of logic clauses. These logic clauses are incorporated into the domain ontology to determine if it contradicts the ontology or not. Our experiments show that it is possible to determine the correctness of sentences with higher precision but lower recall compared with our previous work. Our experiments also show that the performance is dependent on the quality of the extractions of the underlying IE process.

As future goals, we have identified four topics in our work that require improvement. The first topic is the improvement in the identification of entities in the extraction process. In many cases, the entities extracted by the IE process are phrases which cannot be clearly mapped to the concepts of the domain ontology because the extraction includes other elements as part of the entity (e.g., a property). This aspect of IE can have a significant impact in our method because a relationship might become unrelated to the domain if its elements cannot be recognized as part of it. A second topic refers to finding better mechanisms to define mappings between the vocabulary of the text and the vocabulary of the ontology. We believe that this aspect of our method can be automated by the inclusion of text processing methods such as dependency parsing and coreference resolution. The third topic refers to determining the reason why a sentence is incorrect. Although current ontology debugging methods can provide tentative solutions to this problem, they have both different focus and different parameters to find the origin of inconsistency. For example, because in ontology debugging the origin of the inconsistency is not known, a search mechanism must be defined as part of the debugging process. In the case error detection in text the origin of the inconsistency is the analyzed text and the ontological axioms that are related to the analyzed text, so there is no need for a search mechanism.

Finally, the fourth topic refers to the simultaneous analysis of multiple sentences. Because analyzing a set of sentences at once has a significant computational cost, we are considering an incremental approach. Iteratively, we add statements into the ontology and perform consistency checking. If there is an inconsistency, we try to identify the origin. In this approach, a key element is the order of the statements that are being added into the ontology. For example, from a text we can produce the set of statements $S = s_1, \dots, s_i, \dots, s_j, \dots, s_n$ (with i much smaller than j). Let us assume that the inclusion into the ontology of statements s_i and s_j together makes it inconsistent. Then, since i is much smaller than j , in our incremental approach s_j will be added many iterations later after s_i . If we sort the statements with a selection function such as the one in [12], the analysis with both statements can be performed earlier. Although this efficient ordering of statements does not reduce the complexity of the consistency checking, it can reduce the complexity when trying to find the origin of the inconsistency.

Acknowledgments. This research is partially supported by the National Science Foundation grant IIS-1118050 and grant IIS-1013054. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the NSF.

References

1. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., Ant Peter, D.L.M., Patel-Schneider, F., Stein, L.A.: OWL Web Ontology Language, <http://www.w3.org/TR/owl-ref/>
2. Blessing, A., Schütze, H.: Crosslingual distant supervision for extracting relations of different complexity. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM 2012, pp. 1123–1132 (2012)
3. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, pp. 1535–1545 (2011)
4. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. *The Knowledge Engineering Review* 23(02), 117–152 (2008)
5. Gutierrez, F., Dou, D., Fickas, S., Griffiths, G.: Providing Grades and Feedback for Student Summaries by Ontology-based Information Extraction. In: Proceedings of the 21st ACM Conference on Information and Knowledge Management, CIKM 2012, pp. 1722–1726 (2012)
6. Gutierrez, F., Dou, D., Fickas, S., Martini, A., Zong, H.: Hybrid Ontology-based Information Extraction for Automated Text Grading. In: Proceedings of the 12th IEEE International Conference on Machine Learning and Applications, ICMLA 2013, pp. 359–364 (2013)
7. Haase, P., Stojanovic, L.: Consistent evolution of OWL ontologies. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 182–197. Springer, Heidelberg (2005)
8. Haase, P., Völker, J.: Ontology learning and reasoning — dealing with uncertainty and inconsistency. In: da Costa, P.C.G., d'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) URSW 2005 - 2007. LNCS (LNAI), vol. 5327, pp. 366–384. Springer, Heidelberg (2008)
9. Hearst, M.A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: Proceedings of the Fourteenth Conference on Computational Linguistics, COLING 1992, pp. 539–545 (1992)
10. Horridge, M., Parsia, B., Sattler, U.: Explaining inconsistencies in OWL ontologies. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS, vol. 5785, pp. 124–137. Springer, Heidelberg (2009)
11. Horrocks, I., Patel-Schneider, P.F.: Reducing OWL entailment to description logic satisfiability. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 17–29. Springer, Heidelberg (2003)
12. Huang, Z., van Harmelen, F., Teije, A.T.: Reasoning with inconsistent ontologies. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2005, pp. 454–459 (2005)

13. Landauer, T.K., Laham, D., Foltz, P.W.: Learning human-like knowledge by singular value decomposition: a progress report. In: Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10, NIPS 1997, pp. 45–51 (1998)
14. Lin, C.-Y.: Rouge: a package for automatic evaluation of summaries. In: Workshop on Text Summarization Branches Out, pp. 25–26 (2004)
15. Mausam, Schmitz, M., Soderland, S., Bart, R., Etzioni, O.: Open language learning for information extraction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, pp. 523–534 (2012)
16. Miller, G.A.: Wordnet: A lexical database for english. Communications of the ACM 38, 39–41 (1995)
17. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL 2009, pp. 1003–1011 (2009)
18. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. Journal of Artificial Intelligence Research 36, 165–228 (2009)
19. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32(1), 57–95 (1987)
20. Ritter, A., Downey, D., Soderland, S., Etzioni, O.: It's a contradiction—no, it's not: A case study using functional relations. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, pp. 11–20. Association for Computational Linguistics, Stroudsburg (2008)
21. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2003, pp. 355–362 (2003)
22. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. Journal of Automated Reasoning 39(3), 317–349 (2007)
23. Sohlberg, M., Griffiths, G., Fickas, S.: The effect of electronically delivered strategies on reading after mild-moderate acquired brain injury. American Journal of Speech-Language Pathology (November 2011) (in review)
24. Wimalasuriya, D.C., Dou, D.: Ontology-based information extraction: An introduction and a survey of current approaches. Journal of Information Science 36, 306–323 (2010)
25. Wu, F., Weld, D.S.: Autonomously semantifying wikipedia. In: Proceedings of the 16th ACM Conference on Information and Knowledge Management, CIKM 2007, pp. 41–50 (2007)

Making Metaquerying Practical for $Hi(DL-Lite_{\mathcal{R}})$ Knowledge Bases

Maurizio Lenzerini, Lorenzo Lepore, and Antonella Poggi

Dipartimento di Ingegneria Informatica, Automatica e Gestionale “Antonio Ruberti”- Sapienza
Università di Roma
Via Ariosto 25, I-00183 Roma, Italy
lastname@dis.uniroma1.it

Abstract. $Hi(DL-Lite_{\mathcal{R}})$ is a higher-order Description Logic obtained from $DL-Lite_{\mathcal{R}}$ by adding metamodeling features, and is equipped with a query language that is able to express higher-order queries. We investigate the problem of answering a particular class of such queries, called instance higher-order queries, posed over $Hi(DL-Lite_{\mathcal{R}})$ knowledge bases (KBs). The only existing algorithm for this problem is based on the idea of reducing the evaluation of a higher-order query q over a $Hi(DL-Lite_{\mathcal{R}})$ KB to the evaluation of a union of first-order queries over a $DL-Lite_{\mathcal{R}}$ KB, built from q by instantiating its metavariables in all possible ways. Although of polynomial time complexity with respect to the size of the KB, this algorithm turns out to be inefficient in practice. In this paper we present a new algorithm, called Smart Binding Planner (SBP), that compiles Q into a program, that issues a sequence of first-order conjunctive queries, where each query has the goal of providing the bindings for metavariables of the next ones, and the last one completes the process by computing the answers to Q . We also illustrate some experiments showing that, in practice, SBP is significantly more efficient than the previous approach.

1 Introduction

Description Logics (DLs) are popular formalisms for expressing ontologies, where an ontology is regarded as a formal specification of the concepts that are relevant in the domain of interest, together with the relationships between concepts. In many applications, the need arises of modeling and reasoning about metaconcepts and metaproPERTIES. Roughly speaking, a metaconcept is a concept whose instances can be themselves concepts, and a metaproPERTY is a relationship between metaconcepts. Metaconcept representation is needed, for example, in formal ontology, where specific metaproPERTIES (e.g., rigidity) are used to express relevant aspects of the intended meaning of the elements in an ontology. More generally, the idea of representing concepts and properties at the metalevel has been exploited in several sub-fields of Knowledge Representation and Computer Science, including semantic networks, early Frame-based and terminological systems [11,2], and conceptual modeling languages [13]. The notion of metaclass is also present in virtually all object-oriented languages, including modern programming languages (see, for instance, the Java class “class”). Finally, metaclasses can be

modeled in the W3C RDF¹ and RDF Schema (RDFS)² languages, simply because a concept can be modeled as a resource, and resources can be asserted to be instances (`rdf:type`) of other resources.

To see an example of metamodeling, we refer to the Ontology of the Italian Public Debt, which we have designed in a joint project with the Italian Ministry of Economy and Finance [1]. One of the central elements of such an ontology is the concept of financial instrument, which is an abstraction for any contract that gives rise to a financial asset of one entity and a financial liability or equity instrument of another entity. Several subconcepts of financial instrument, such as “Buono del Tesoro Poliennale” (or simply “BTP”, which is the Italian bond considered as benchmark for measuring the Italian spread) or “commercial paper”, are of interest in the domain. We are also interested in various properties of financial instruments, including the duration, the entity for which it produces an asset, and the type (e.g., “BTP”, or “commercial paper”). For example, the financial instrument “ib_03_121” has type “BTP” and 5 years duration, and “jh_09_78” has type “commercial paper” and 5 years duration. Note that, if a financial instrument has type “T”, then it is an instance of “T”; so, types are both individuals having relationships with other individuals, and concepts. Besides properties of financial instruments, users are interested on properties of types of financial instruments, such as the financial institution (e.g., Bank of Italy) issuing the corresponding financial instruments, and the category to which the type belongs. To model such types and specify their properties, the ontology contains the concept “type of financial instrument”, whose instances are the relevant types in the domain. Thus, the concept “BTP” is an instance of the concept “type of financial instrument”. More precisely, “BTP” is an instance of “type of Italian financial instrument” which is a subconcept of “type of financial instrument” (another such subconcept is “type of foreign financial instruments”).

For every type of financial instrument that is modeled as a concept in the ontology (e.g., “type of Italian financial instrument”), users are also interested in the amount of public debt cumulatively generated by the financial instruments of that type. This is modeled as a property of the concept “category of financial instrument type”, whose instances are exactly the types of financial instruments in the ontology. The various subconcepts of “category of financial instrument” model different sets of types of financial instruments based on specific classification criteria. For example, “category of financial instrument with respect to market” is a subconcept of “category of financial instrument”, and its various instances are types of financial instruments differing for the kind of market of issuance (e.g., “type of Italian financial instruments” vs. “type of foreign financial instruments”). Another example of subconcept of “category of financial instrument” is the concept “category of financial instrument with respect to payment mode”, whose instances differ for the payment mode associated with the financial instruments (e.g., with or without coupons).

The diagrammatic representation of the ontology appears in Figure 1, expressed in the visual language *Graphol* [7]. In the diagram, thick arrows with no labels denote inclusion (i.e., is-a) assertions, thick arrows with labels denote instance-of assertions, and suitable graphical symbols denote expressions. For the sake of simplicity, disjointness

¹ <http://www.w3.org/RDF/>

² <http://www.w3.org/TR/rdf-schema/>

assertions are not shown. For every ontology element e occurring within an assertion, if it occurs (i) as concept argument, it is graphically represented as a rectangle, (ii) as role argument, it is represented as a diamond, and, finally, (iii) neither as concept nor as role argument, it is represented as an octagon.

Notice that the above ontology cannot be modeled in RDF. Indeed, the assertions stating that every instance of a concept participate in at least one instance of a property (e.g., every type of financial instrument has an issuer) is not expressible neither in RDF nor in RDFS. On the other hand, we will see in Section 2 that the ontology fragment described in Figure 1 can be expressed in an extension of the Description Logic $DL\text{-}Lite}_{\mathcal{R}}$ [5].

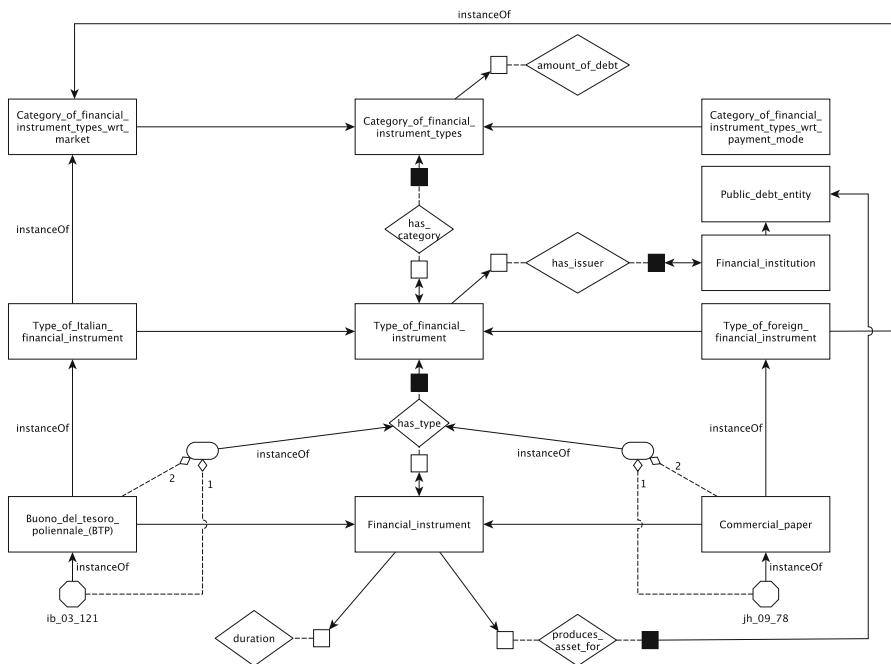


Fig. 1. Diagrammatic representation of KB in Table 1

Obviously, the benefit of using metamodeling is greatly limited if one cannot express metaqueries on the knowledge base. A metaquery is an expression that combines predicates and metapredicates in order to extract complex patterns from the knowledge base. In particular, in a metaquery, variables can appear in predicate positions. In the example above, one might be interested in asking for all financial instruments that are instances of a type that is an instance of 'Category of financial instrument types wrt market', thus traversing three levels of the instance-of relation.

It is well-known that in logic, higher-order constructs are needed for a correct representation of concepts and properties at the meta-level. However, current research on

Description Logics only rarely addresses the issue of extending the language with higher-order constructs (see, for instance, [3,6,12,14,10]). The starting point of our investigation is the work presented in [8,9], where the authors propose a method to add medamodeling facilities to Description Logics, and study algorithms for answering metaqueries in a particular logic, called $Hi(DL-Lite_{\mathcal{R}})$. The logic $Hi(DL-Lite_{\mathcal{R}})$ is obtained from $DL-Lite_{\mathcal{R}}$ [5] by adding suitable constructs for metaconcepts and metaproPERTIES modeling. Since the logics of the $DL-Lite$ family are equipped with query answering algorithms with nice computational properties with respect to the size of the data, we believe that $Hi(DL-Lite_{\mathcal{R}})$ is a good candidate for a language that can be used in ontology-based data access applications requiring to model metaconcepts and metaproPERTIES.

It was shown in [8] that answering higher-order unions of conjunctive queries (HUCQs) over $Hi(DL-Lite_{\mathcal{R}})$ KBs is in general intractable, while answering instance-based HUCQ (IHUCQ) has the same complexity as answering standard (i.e., first-order) unions of conjunctive queries over $DL-Lite_{\mathcal{R}}$ KBs.

In particular, in [8] the authors present a query answering technique based on the idea of first transforming an IHUCQ Q over a $Hi(DL-Lite_{\mathcal{R}})$ KBs \mathcal{H} into a metaground IHUCQ Q' and then computing the certain answers to Q' over \mathcal{H} seen as a standard $DL-Lite_{\mathcal{R}}$ KB. Note that a metaground IHUCQ has only constants in predicate positions, and therefore is a first-order query. The goal of the technique was essentially to show that the data complexity class of query answering does not change if we add metamodeling to $DL-Lite_{\mathcal{R}}$. However, the technique presented in the paper turns out to be impractical in real world cases. Indeed, in most cases, the query answering algorithm computes the union of a huge number of metaground IHCQs, most of which can be in fact empty. This is due to the fact that the metagrounding is computed “blindly”, i.e., by essentially instantiating in all possible ways all variables with the elements that are in predicate positions.

In this paper, we present a new algorithm for answering IHUCQs posed over $Hi(DL-Lite_{\mathcal{R}})$ knowledge bases, called *Smart Binding Planner (SBP)*. The basic idea of the algorithm is to compile the query q into a program expressed in a Datalog-like language, that issues a sequence of first-order conjunctive queries, where each query has the goal of providing the bindings for meta-predicates of the next ones, and the last one completes the process by computing the answers to q . We also illustrate some preliminary experiments showing that, in practice, SBP is significantly more efficient than the previous approach.

The paper is organized as follows. In Section 2 we describe the $Hi(DL-Lite_{\mathcal{R}})$ logic. In Section 3 we discuss the problem of expressing and evaluating IHUCQs in $Hi(DL-Lite_{\mathcal{R}})$, and we briefly illustrate the algorithm presented in [8]. In Section 4 we present our new algorithm, and Section 5 describes a set of experiments aiming at illustrating the behavior of our algorithm and at comparing it with the previous technique. Section 6 concludes the paper.

2 $Hi(DL-Lite_{\mathcal{R}})$

In the following we recall the basics of $Hi(DL-Lite_{\mathcal{R}})$ [8], by first presenting its syntax and then its semantics. Following [8], we can characterize a traditional DL \mathcal{L} by a set

$OP(\mathcal{L})$ of *operators*, used to form concept and role expressions, and a set of $MP(\mathcal{L})$ of *metapredicates*, used to form assertions. Each operator and each metapredicate have an associated arity. If symbol S has arity n , then we write S/n to denote such a symbol and its arity. For $DL\text{-}Lite_{\mathcal{R}}$, we have: (i) $OP(DL\text{-}Lite_{\mathcal{R}}) = \{Inv/1, Exists/1\}$, which stand for “inverse role”, and “existential restriction”, (ii) $MP(DL\text{-}Lite_{\mathcal{R}}) = \{Inst_C/2, Inst_R/3, Isa_C/2, Isa_R/2, Disj_C/2, Disj_R/2\}$, which stand for instance, isa and disjoint assertions on both concepts and roles.

Syntax. Given a countably infinite alphabet \mathcal{S} of *element names*, we inductively define the set of *expressions* for $Hi(DL\text{-}Lite_{\mathcal{R}})$, denoted by $\mathcal{E}_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S})$, over the alphabet \mathcal{S} as follows:

- if $e \in \mathcal{S}$, then $e \in \mathcal{E}_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S})$;
- if $e \in \mathcal{E}_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S})$, and e is not of the form $Inv(e')$ (where e' is any expression), then $Inv(e) \in \mathcal{E}_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S})$;
- if $e \in \mathcal{E}_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S})$, then $Exists(e) \in \mathcal{E}_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S})$.

Intuitively, expressions denote elements, i.e., individuals, concepts and roles, of the knowledge base. The names in \mathcal{S} are the symbols denoting the atomic elements, while the expressions denote either atomic elements, inverses of atomic elements (when interpreted as roles), or projections of atomic elements on either the first or the second component (again, when such elements are interpreted as roles).

The basic building blocks of a $Hi(DL\text{-}Lite_{\mathcal{R}})$ knowledge base are assertions. A $DL\text{-}Lite_{\mathcal{R}}$ -*assertion*, or simply *assertion*, over the alphabet \mathcal{S} for $Hi(DL\text{-}Lite_{\mathcal{R}})$ is a statement of one of the forms

$$a_1(e_1, e_2), a_2(e_1, e_2, e_3)$$

where $a_1 \in MP(DL\text{-}Lite_{\mathcal{R}})$ is either $Inst_C, Isa_C, Isa_R, Disj_C, Disj_R$, a_2 is $Inst_R$, and e_1, e_2, e_3 are expressions in $\mathcal{E}_{DL\text{-}Lite_{\mathcal{R}}}(\mathcal{S})$. Thus, an assertion is simply an application of a metapredicate to a set of expressions, which intuitively means that an assertion is an axiom that predicates over a set of individuals, concepts or roles. A $Hi(DL\text{-}Lite_{\mathcal{R}})$ knowledge base (*KB*) over \mathcal{S} is a finite set of $Hi(DL\text{-}Lite_{\mathcal{R}})$ -assertions over \mathcal{S} .

Example 1. We refer to the example discussed in the introduction, and show the corresponding $Hi(DL\text{-}Lite_{\mathcal{R}})$ knowledge base in Table 1.

Before tackling, in the next section, query answering, we follow [9], and introduce some notions that will be used in the next sections. We say that:

- in the assertion $Inst_C(e, e')$, e occurs as object argument, and e' occurs as a concept argument,
- in the assertion $Inst_R(e, e', e'')$, e, e' occur as object arguments, and e'' occurs as a role argument,
- in the assertion $Isa_C(e, e')$, e, e' occur as concept arguments,
- in the assertion $Isa_R(e, e')$, e, e' occur as role arguments,
- in the assertion $Disj_C(e, e')$, e, e' occur as concept arguments,
- in the assertion $Disj_R(e, e')$, e, e' occur as role arguments.

Table 1. $Hi(DL-Lite_{\mathcal{R}})$ KB

$Inst_C(ib_03_121, BTP)$	$Inst_C(Type_of_Italian_f_i, Category_of_f_i_wrt_market)$
$Inst_C(BTP, Type_of_Italian_f_i)$	$Inst_C(Commercial_paper, Type_of_foreign_f_i)$
$Inst_C(jh_09_78, Commercial_paper)$	$Inst_C(Type_of_foreign_f_i, Category_of_f_i_wrt_market)$
$Inst_R(ib_03_121, BTP, has_type)$	$Inst_R(jh_09_78, Commercial_paper, has_type)$
$Inst_R(ib_03_121, 5, duration)$	$Inst_R(jh_09_78, 5, duration)$
$Isa_C(BTP, Financial_instrument)$	$Isa_C(Exists(Inv(has_category)), Category_of_f_i_types)$
$Isa_C(Financial_instrument, Exists(duration))$	$Isa_C(Financial_instrument, Exists(produces_asset_for))$
$Isa_C(Financial_institution, Public_debt_entity)$	$Isa_C(Exists(Inv(produces_asset_for)), Public_debt_entity)$
$Isa_C(Financial_instrument, Exists(has_type))$	$Isa_C(Type_of_Italian_f_i, Type_of_financial_instrument)$
$Isa_C(Exists(has_type), Financial_instrument)$	$Isa_C(Type_of_foreign_f_i, Type_of_financial_instrument)$
$Isa_C(Type_of_financial_instrument, Exists(has_issuer))$	$Isa_C(Type_of_financial_instrument, Exists(has_category))$
$Isa_C(Commercial_paper, Financial_instrument)$	$Isa_C(Exists(has_category), Type_of_financial_instrument)$
$Isa_C(Category_of_f_i_types, Exists(amount_of_debt))$	$Isa_C(Category_of_f_i_wrt_pay_mode, Category_of_f_i_types)$
$Isa_C(Exists(Inv(has_issuer)), Financial_institution)$	$Isa_C(Category_of_f_i_wrt_market, Category_of_f_i_types)$
$Isa_C(Financial_institution, Exists(Inv(has_issuer)))$	
$Disj_C(BTP, Commercial_paper)$	$Disj_C(Type_of_Italian_f_i, Type_of_foreign_f_i)$

Semantics. The semantics of $Hi(DL-Lite_{\mathcal{R}})$ is based on the notion of interpretation structure. An *interpretation structure* is a triple $\Sigma = \langle \Delta, \mathcal{I}_C, \mathcal{I}_R \rangle$ where: (i) Δ is a non-empty (possibly countably infinite) set; (ii) \mathcal{I}_C is a function that maps each $d \in \Delta$ into a subset of Δ ; and (iii) \mathcal{I}_R is a function that maps each $d \in \Delta$ into a subset of $\Delta \times \Delta$. In other words, Σ treats every element of Δ simultaneously as: (i) an individual; (ii) a unary relation, i.e., a concept, through \mathcal{I}_C ; and (iii) a binary relation, i.e., a role, through \mathcal{I}_R . An *interpretation* for \mathcal{S} (simply called an interpretation, when \mathcal{S} is clear from the context) over the interpretation structure Σ is a pair $\mathcal{I} = \langle \Sigma, \mathcal{I}_o \rangle$, where

- $\Sigma = \langle \Delta, \mathcal{I}_C, \mathcal{I}_R \rangle$ is an interpretation structure, and
- \mathcal{I}_o is the interpretation function of \mathcal{I} , i.e., a function such that:
 1. \mathcal{I}_o maps each $op \in OP(DL-Lite_{\mathcal{R}})$ to a function $op^{\mathcal{I}_o} : \Delta \rightarrow \Delta$ that satisfies the conditions characterizing the operator op . In particular, the conditions for the operators in $OP(DL-Lite_{\mathcal{R}})$ are as follows:
 - (a) for each $d_1 \in \Delta$, if $d = Inv^{\mathcal{I}_o}(d_1)$ then $d^{\mathcal{I}_r} = (d_1^{\mathcal{I}_r})^{-1}$;
 - (b) for each $d_1 \in \Delta$, if $d = Exists^{\mathcal{I}_o}(d_1)$ then $d^{\mathcal{I}_c} = \{d \mid \exists d' \text{ s.t. } \langle d, d' \rangle \in d_1^{\mathcal{I}_R}\}$.
 2. \mathcal{I}_o maps each $e \in \mathcal{E}_{DL-Lite_{\mathcal{R}}}(\mathcal{S})$ to Δ as follows:
 - (a) if $e \in \mathcal{S}$, then simply $e^{\mathcal{I}_o} \in \Delta$;
 - (b) if e is of the form $Inv(e_1)$, then $e^{\mathcal{I}_o} = Inv^{\mathcal{I}_o}(e_1^{\mathcal{I}_o})$;
 - (c) if e is of the form $Exists(e_1)$, then $e^{\mathcal{I}_o} = Exists^{\mathcal{I}_o}(e_1^{\mathcal{I}_o})$.

Finally, we define the semantics of assertions, by defining the notion of satisfaction of an assertion with respect to an interpretation \mathcal{I} as follows:

- $\mathcal{I} \models Inst_C(e_1, e_2)$ if $e_1^{\mathcal{I}_o} \in (e_2^{\mathcal{I}_o})^{\mathcal{I}_c}$;
- $\mathcal{I} \models Inst_R(e_1, e_2, e_3)$ if $\langle e_1^{\mathcal{I}_o}, e_2^{\mathcal{I}_o} \rangle \in (e_3^{\mathcal{I}_o})^{\mathcal{I}_r}$;
- $\mathcal{I} \models Isa_C(e_1, e_2)$ if $(e_1^{\mathcal{I}_o})^{\mathcal{I}_c} \subseteq (e_2^{\mathcal{I}_o})^{\mathcal{I}_c}$;
- $\mathcal{I} \models Isa_R(e_1, e_2)$ if $(e_1^{\mathcal{I}_o})^{\mathcal{I}_r} \subseteq (e_2^{\mathcal{I}_o})^{\mathcal{I}_r}$;
- $\mathcal{I} \models Disj_C(e_1, e_2)$ if $(e_1^{\mathcal{I}_o})^{\mathcal{I}_c} \cap (e_2^{\mathcal{I}_o})^{\mathcal{I}_c} = \emptyset$;
- $\mathcal{I} \models Disj_R(e_1, e_2)$ if $(e_1^{\mathcal{I}_o})^{\mathcal{I}_r} \cap (e_2^{\mathcal{I}_o})^{\mathcal{I}_r} = \emptyset$.

A $Hi(DL\text{-}Lite}_{\mathcal{R}}$) KB \mathcal{H} is satisfied by \mathcal{I} if all the assertions in \mathcal{H} are satisfied by \mathcal{I} . As usual, the interpretations \mathcal{I} satisfying \mathcal{H} are called the *models* of \mathcal{H} . A $Hi(DL\text{-}Lite}_{\mathcal{R}}$) KB \mathcal{H} is *satisfiable* if it has at least one model.

3 Querying $Hi(DL\text{-}Lite}_{\mathcal{R}}$) Knowledge Bases

In this section we address the issue of querying $Hi(DL\text{-}Lite}_{\mathcal{R}}$) knowledge bases. Specifically, we define queries over $Hi(DL\text{-}Lite}_{\mathcal{R}}$) knowledge bases and report on the only query answering technique over such knowledge bases we are aware of. In the next section, we will then propose our new technique for query answering.

In order to define $Hi(DL\text{-}Lite}_{\mathcal{R}}$) queries, we first need to introduce the notion of “atom”. We consider a countably infinite alphabet of variables \mathcal{V} , disjoint from \mathcal{S} , and a countably infinite alphabet of query predicates, each with its arity, disjoint from all other alphabets. An *atom* over \mathcal{S} and \mathcal{V} (or simply, an atom) has the form $a_1(e_1, e_2)$, $a_2(e_1, e_2, e_3)$ where $a_1 \in MP(DL\text{-}Lite}_{\mathcal{R}}$ is either $Inst_C$, Isa_C , Isa_R , $Disj_C$ or $Disj_R$, a_2 is $Inst_R$, and each e_i is either an expression in $\mathcal{E}_{DL\text{-}Lite}_{\mathcal{R}}(\mathcal{S})$ or a variable in \mathcal{V} , i.e., $e_i \in \mathcal{E}_{DL\text{-}Lite}_{\mathcal{R}}(\mathcal{S}) \cup \mathcal{V}$. In the above assertions, a_1 and a_2 are called the predicates of the atom.

A *higher-order conjunctive query (HCQ)* of arity n is a formula of the form $q(u_1, \dots, u_n) \leftarrow a_1, \dots, a_m$ where $n \geq 0$, $m \geq 1$, q is a query predicate of arity n (which is also the arity of the query), each a_i is an atom, and each u_i is either in \mathcal{V} and occurs in some a_j , or it is in $\mathcal{E}_{DL\text{-}Lite}_{\mathcal{R}}(\mathcal{S})$. As usual, the tuple $\mathbf{u} = (u_1, \dots, u_n)$ is called the *target of the query* q and the variables in \mathbf{u} are its *distinguished variables*. The other variables of the query are called *existential variables*. Also, similarly to names within assertions, we say that a variable *occurs as concept argument or role argument within an atom*, depending on its position. Variables occurring as concept or role arguments are called *metavariables*. A *higher-order union of conjunctive queries (HUCQ)* of arity n is a set of HCQs of arity n with the same query predicate. Also a HUCQ is *metaground* if it does not contain any metavariable.

A HCQ is called *instance higher-order conjunctive query (IHCQ)* if each of its atoms has $Inst_C$ and $Inst_R$ as predicate. A HUCQ containing only IHCQs is called *instance higher-order union of conjunctive queries (IHUCQ)*.

Finally, a higher-order query is called *Boolean* if it has arity 0.

Example 2. Consider the $Hi(DL\text{-}Lite}_{\mathcal{R}}$) KB \mathcal{H} of Example 1. Suppose one needs to retrieve all pairs of financial instruments having the same duration, such that one is Italian (i.e., its type is an instance of “type of Italian financial instrument”), and is issued by a Public Debt entity, and the other one is foreign (i.e., its type is an instance of “type of foreign financial instrument”). The IHCQ expressing this need is the following:

$$\begin{aligned} q_7(x, y) \leftarrow & Inst_R(x, z, \text{duration}), Inst_R(y, z, \text{duration}), Inst_C(x, v), \\ & Inst_C(v, \text{Type_of_italian_f_i}), Inst_R(v, t, \text{has_issuer}), \\ & Inst_C(t, \text{Public_debt_entity}), Inst_C(y, w), Inst_C(w, \text{Type_of_foreign_f_i}). \end{aligned}$$

□

We observe that, although both the concept “Public_debt_entity”, and the role “has_issuer” are empty in \mathcal{H} , nevertheless, the set of certain answers to the query q_7 of the above example is $\{\langle "ib_03_121", "jh_09_78", 5 \rangle\}$. Indeed, reasoning about the query and the ontology allows concluding that the two atoms $Inst_R(w, t, \text{has_issuer})$ and $Inst_C(t, \text{Public_debt_entity})$ are redundant in the query (since the ontology sanctions that every type of financial instrument is issued by at least one financial institution, and every financial institution is a public debt entity). We point out that this kind of reasoning is not supported by RDF(S)-based reasoners.

In order to define the semantics of queries, we now introduce the notion of assignment. Indeed, to interpret non-ground terms, we need assignments over interpretations, where an *assignment* μ over $\langle \Sigma, \mathcal{I}_o \rangle$ is a function $\mu : \mathcal{V} \rightarrow \Delta$. Given an interpretation $\mathcal{I} = \langle \Sigma, \mathcal{I}_o \rangle$ and an assignment μ over \mathcal{I} , the interpretation of terms is specified by the function $(\cdot)^{\mathcal{I}_o, \mu} : \tau_{DL-Lite_{\mathcal{R}}}(\mathcal{S}, \mathcal{V}) \rightarrow \Delta$ defined as follows: (i) if $e \in \mathcal{S}$ then $e^{\mathcal{I}_o, \mu} = e^{\mathcal{I}_o}$; (ii) if $e \in \mathcal{V}$ then $e^{\mathcal{I}_o, \mu} = \mu(e)$; (iii) $op(e)^{\mathcal{I}_o, \mu} = op^{\mathcal{I}_o}(e^{\mathcal{I}_o, \mu})$.

Finally, we define the semantics of atoms, by defining the notion of satisfaction of an atom with respect to an interpretation \mathcal{I} and an assignment μ over \mathcal{I} as follows:

- $\mathcal{I}, \mu \models Inst_C(e_1, e_2)$ if $e_1^{\mathcal{I}_o, \mu} \in (e_2^{\mathcal{I}_o, \mu})^{\mathcal{I}_c}$;
- $\mathcal{I}, \mu \models Inst_R(e_1, e_2, e_3)$ if $(e_1^{\mathcal{I}_o, \mu}, e_2^{\mathcal{I}_o, \mu}) \in (e_3^{\mathcal{I}_o, \mu})^{\mathcal{I}_r}$;
- $\mathcal{I}, \mu \models Isa_C(e_1, e_2)$ if $(e_1^{\mathcal{I}_o, \mu})^{\mathcal{I}_c} \subseteq (e_2^{\mathcal{I}_o, \mu})^{\mathcal{I}_c}$;
- $\mathcal{I}, \mu \models Isa_R(e_1, e_2)$ if $(e_1^{\mathcal{I}_o, \mu})^{\mathcal{I}_r} \subseteq (e_2^{\mathcal{I}_o, \mu})^{\mathcal{I}_r}$;
- $\mathcal{I}, \mu \models Disj_C(e_1, e_2)$ if $(e_1^{\mathcal{I}_o, \mu})^{\mathcal{I}_c} \cap (e_2^{\mathcal{I}_o, \mu})^{\mathcal{I}_c} = \emptyset$;
- $\mathcal{I}, \mu \models Disj_R(e_1, e_2)$ if $(e_1^{\mathcal{I}_o, \mu})^{\mathcal{I}_r} \cap (e_2^{\mathcal{I}_o, \mu})^{\mathcal{I}_r} = \emptyset$.

Let \mathcal{I} be an interpretation and μ an assignment over \mathcal{I} . A Boolean HCQ q of the form $q \leftarrow a_1, \dots, a_n$ is *satisfied* in \mathcal{I}, μ if every query atom a_i is satisfied in \mathcal{I}, μ . Given a Boolean HCQ q and a $Hi(DL-Lite_{\mathcal{R}})$ KB \mathcal{H} , we say that q is *logically implied* by \mathcal{H} (denoted by $\mathcal{H} \models q$) if for each model \mathcal{I} of \mathcal{H} there exists an assignment μ such that q is satisfied by \mathcal{I}, μ . Given a non-Boolean HCQ q of the form $q(e_1, \dots, e_n) \leftarrow a_1, \dots, a_m$, a grounding substitution of q is a substitution θ such that $e_1\theta, \dots, e_n\theta$ are ground terms. We call $e_1\theta, \dots, e_n\theta$ a grounding tuple. The set of *certain answers* to q in \mathcal{H} , denoted by $cert(q, \mathcal{H})$, is the set of grounding tuples $e_1\theta, \dots, e_n\theta$ that make the Boolean query $q_\theta \leftarrow a_1\theta, \dots, a_n\theta$ logically implied by \mathcal{H} . These notions extend immediately to HUCQs.

As we said in the introduction, in [8] the authors present a query answering technique based on the idea of first transforming a IHUCQ Q over a $Hi(DL-Lite_{\mathcal{R}})$ KBs \mathcal{H} into a metaground IHUCQ Q' and then computing the certain answers to Q' over \mathcal{H} seen as a standard $DL-Lite_{\mathcal{R}}$ KB³. It was shown that query answering with such a technique is in AC⁰ w.r.t. the number of instance assertions, in PTIME w.r.t. KB complexity, and NP-complete w.r.t. combined complexity. While describing, in the following, the technique of [8] in more details, we introduce notions that we use in the next sections.

Given a $Hi(DL-Lite_{\mathcal{R}})$ KB \mathcal{H} , we respectively denote by $Roles(\mathcal{H})$ and $Concepts(\mathcal{H})$ the sets $Roles(\mathcal{H}) = \{e, Inv(e) | e \in \mathcal{E}_{DL-Lite_{\mathcal{R}}}(\mathcal{S}) \text{ and } e \text{ occurs as role}$

³ In fact, in [8], the authors consider KBs equipped with mappings, and therefore their technique aims at rewriting the initial query over the actual data sources. Here, we ignore this aspect and adapt their algorithm to our scenario.

argument in \mathcal{H} } and $Concepts(\mathcal{H}) = \{e \mid e \in \mathcal{E}_{DL-Lite_{\mathcal{R}}}(\mathcal{S}) \text{ and } e \text{ occurs as concept argument in } \mathcal{H}\} \cup \{\exists e (\text{Exists}(e), \text{Exists}(\text{Inv}(e)) \mid e \in \mathcal{E}_{DL-Lite_{\mathcal{R}}}(\mathcal{S}) \text{ and } e \text{ occurs as role argument in } \mathcal{H}\}$. Given two IHCQs q, q' and a KB \mathcal{H} , we say that q' is a *partial metagrounding of q with respect to \mathcal{H}* if $q' = \sigma(q)$ where σ is a partial substitution of the metavariables of q such that for each metavariable x of q , either $\sigma(x) = x$, or: (i) if x occurs in concept position in q , then $\sigma(x) \in Concepts(\mathcal{H})$; (ii) if x occurs in role position in q , then $\sigma(x) \in Roles(\mathcal{H})$. Given a IHCQ q and a KB \mathcal{H} , we denote by $PMG(q, \mathcal{H})$ the IHUCQ constituted by the union of all the partial metagroundings of q w.r.t. \mathcal{H} . Moreover given a IHUCQ Q and a KB \mathcal{H} , we denote by $PMG(Q, \mathcal{H})$ the IHUCQ formed by $\bigcup_{q \in Q} PMG(q, \mathcal{H})$.

With these notions in place, the algorithm of [8], which we denote as COMPUTEPMG, can be described as follows. Given an IHUCQ Q and a $Hi(DL-Lite_{\mathcal{R}})$ KB \mathcal{H} , it first computes the query $PMG(Q, \mathcal{H})$ and then returns the certain answers to the metaground query $PMG(Q, \mathcal{H})$ with respect to \mathcal{H} . It is worth noting that, although polynomial w.r.t. the number of instance assertions in \mathcal{H} , COMPUTEPMG can be very inefficient in practice, in the sense that it computes the union of a huge number of metaground IHCQs, most of which can be in fact empty. This is due to the fact that the partial metagrounding is computed “blindly”, i.e., by instantiating in all possible ways, all metavariables with elements of $Concepts(\mathcal{H})$ and $Roles(\mathcal{H})$.

4 New Algorithm for Answering IHUCQs in $Hi(DL-Lite_{\mathcal{R}})$

In this section we present the *Smart Binding Planner* (SBP) algorithm, aiming at reducing the number of metaground queries to be evaluated. The algorithm is based on a process that computes a sequence of metaground IHCQs, where each query has the goal of providing both the answers to the query, and the bindings for metavariables of the next ones.

Before delving into the details of the SBP algorithm, we shortly sketch its three main steps. First, it splits the query into a sequence of subqueries (function SPLITANDORDER) such that the evaluation of the i -th subquery provides the bindings to instantiate the metavariables of the $(i + 1)$ -th subquery. Second, based on such subqueries ordering, it builds a program (function DHQPSYNTHESIS), expressed in a specific language named *Datalog-based Higher Order Query Plan* (DHQP). Then, as third and final step, it evaluates the DHQP program (function EVALUATEDHQP).

In the following, we start by presenting the DHQP language and illustrating the function EVALUATEDHQP. We then define the two other functions that are used by SBP, namely the functions SPLITANDORDER and DHQPSYNTHESIS, and finally we present the complete algorithm SBP, and discuss its properties.

4.1 The DHQP Language

Let \mathcal{A}_Γ and \mathcal{A}_I be two pairwise disjoint alphabets, disjoint from \mathcal{S} and \mathcal{V} , called, respectively, the alphabet of *bridge predicates* and the alphabet of *intensional predicates*, each one with an associated arity. Intensional and bridge predicates are used to form a DHQP program, which, intuitively, is constituted by an ordered set of Datalog-like rules

whose head predicate is an intensional predicate in \mathcal{A}_I , and whose body contains an atom with a bridge predicate in \mathcal{A}_{Γ} . Every bridge predicate γ of arity n has an associated IHCQ of arity n , denoted $\text{def}(\gamma)$. Moreover, some of the n arguments of each bridge predicate γ are classified as “to-be-bound”. When we write a bridge predicate, we indicate the variables corresponding to such arguments by underlining them. Intuitively, the underlined variables of the bridge predicate γ are those variables that are to be bound to ground expressions whenever we want to evaluate the query $\text{def}(\gamma)$. Also, the bridge predicate is used to denote the relation where we store the certain answers of a set of metaground IHCQs, each one obtained by $\text{def}(\gamma)$ by substituting the underlined variables of γ with a ground expression. Variables of the query $\text{def}(\gamma)$ that appear underlined in the corresponding bridge predicate γ are called *input variables* of the query. Note that, when we write the query predicate of $\text{def}(\gamma)$, we underline the query input variables.

We now provide the definition of the syntax of a DHQP program. Let m be a non-negative integer, let Γ be a sequence $(\gamma_0, \dots, \gamma_m)$ of $m+1$ bridge predicates in \mathcal{A}_{Γ} , and let I be a sequence (I_0, \dots, I_m) of $m+1$ intensional predicates in \mathcal{A}_I .

A DHQP program P over Γ and I is a sequence of $m+1$ DHQP rules r_0, r_1, \dots, r_m , such that

- r_0 , called the *base rule* of P , has the form $I_0(v) \leftarrow \gamma_0(w)$, where every variable in v occurs in w .
- for each $1 \leq j \leq m$, rule r_j has the form $I_j(v) \leftarrow I_{j-1}(u), \gamma_j(w)$, where every variable in v occurs in u or w , and every variable in u which is not an underlined variable in w appears in v .

For every $0 \leq j \leq m$, we say that r_j defines I_j using γ_j . In the case of $1 \leq j \leq m$, we say that r_j defines I_j using γ_j based on I_{j-1} .

Example 3. Let $I = (I_0/2, I_1/3)$, and $\Gamma = (\gamma_0/2, \gamma_1/5)$, with q_{γ_0} and q_{γ_1} defined as follows:

$$\begin{aligned} q_{\gamma_0}(v, w) &= \text{Inst}_C(v, \text{Type_of_Italian_f_i}), \text{Inst}_C(w, \text{Type_of_foreign_f_i}), \\ &\quad \text{Inst}_R(v, t, \text{has_issuer}), \text{Inst}_C(t, \text{Public_debt_entity}) \\ q_{\gamma_1}(\underline{v}, \underline{w}, x, y, z) &= \text{Inst}_C(x, v), \text{Inst}_C(y, w), \text{Inst}_R(x, z, \text{duration}), \\ &\quad \text{Inst}_R(y, z, \text{duration}). \end{aligned}$$

The following is a DHQP program over Γ and I , called P :

$$\begin{aligned} r_0 : I_0(v, w) &\leftarrow \gamma_0(v, w) \\ r_1 : I_1(x, y, z) &\leftarrow I_0(v, w), \gamma_1(\underline{v}, \underline{w}, x, y, z) \end{aligned}$$

□

We now turn our attention to the semantics of a DHQP program. Toward this goal, we introduce the notion of *instantiation of an IHCQ*. Given an IHCQ q , an n -tuple \mathbf{x} of variables of q , and an n -tuple \mathbf{t} of expressions in $\mathcal{E}_{DL-Lite_{\mathcal{R}}}(\mathcal{S})$, the \mathbf{x} -instantiation of q with \mathbf{t} , denoted $\text{INST}(q, \mathbf{x} \leftarrow \mathbf{t})$, is the IHCQ obtained from q by substituting each x_i in \mathbf{x} with the expression t_i in \mathbf{t} , for $i \in \{1, \dots, n\}$. Note that a partial metagrounding of q with respect to \mathcal{H} is an \mathbf{x} -instantiation of q with any n -tuple $\mathbf{t} = (t_1, t_2, \dots, t_n)$, where $\mathbf{x} = (x_1, x_2, \dots, x_n)$ contains only metavariables in q , and such that, for every

$i \in \{1, \dots, n\}$, $t_i \in \text{Concept}(\mathcal{H})$ if x_i occurs as concept argument within q , and $t_i \in \text{Role}(\mathcal{H})$ if x_i occurs as role argument.

Let P be a DHQP program constituted by the rules (r_0, \dots, r_m) . We specify the semantics of P operationally, i.e., we define an algorithm, called EVALUATEDHQP, that, given as input a DHQP program P , and a $\text{Hi}(DL\text{-Lite}_{\mathcal{R}})$ KB \mathcal{H} , computes the result of evaluating P with respect to a \mathcal{H} as follows:

1. Consider rule r_0 , and compute the extension of I_0 as the certain answers to the metaground query $PMG(q_{\gamma_0}, \mathcal{H})$ over the KB \mathcal{H} .
2. For each $1 \leq i \leq m$, consider rule r_i , and compute the extension of I_i by evaluating the Datalog rule $I_i(\mathbf{v}) \leftarrow I_{i-1}(\mathbf{u}), \gamma_i(\mathbf{w})$, where the extension of I_{i-1} is the result computed when considering rule r_{i-1} , and the extension of γ_i is constituted by the certain answers of the metaground query $PMG(q', \mathcal{H})$ over the KB \mathcal{H} , with q' constructed as follows: $\bigcup_{t \in \pi_{\mathbf{w}'}(I_{i-1})} \text{INST}(q_{\gamma_i}, \mathbf{w}' \leftarrow t)$, where \mathbf{w}' denotes the variables that appear both in \mathbf{u} and in \mathbf{w} , and $\pi_{\mathbf{w}'}(R)$ denotes the projection of relation R onto the arguments corresponding to \mathbf{w}' .
3. Return the extension of the predicate I_m as the result of evaluating P over \mathcal{H} .

Example 4. Consider the DHQP program P described in Example 3, and the KB \mathcal{H} described in Example 1. The algorithm EVALUATEDHQP(P, \mathcal{H}) proceeds as follows. First, it considers rule r_0 and computes the certain answers of the metaground query q_{γ_0} with respect to \mathcal{H} , which, in this case, produces the following result: $\{\langle (\text{BTP}, \text{Commercial_paper}) \rangle\}$. By means of the rule r_0 , this result becomes also the extension of I_0 . Then, it considers rule r_1 , and, for each tuple of I_0 (in this case, only one) evaluates the query resulting from the $\langle v, w \rangle$ -instantiation of q_{γ_1} with the tuple itself; this means that the certain answers to the metaground query

$$\begin{aligned} q(x, y, z) \leftarrow & \text{Inst}_C(x, \text{BTP}), \text{Inst}_C(y, \text{Commercial_paper}), \text{Inst}_R(x, z, \text{duration}), \\ & \text{Inst}_R(y, z, \text{duration}) \end{aligned}$$

are computed and returned as final result, which in this case is $\{(ib_03_121, jh_09_78, 5)\}$. \square

4.2 The Function SPLITANDORDER

Given an IHCQ q with distinguished variables x , SPLITANDORDER returns a sequence of subqueries of q by proceeding as follows.

1. It builds the *dependency graph* of q , which is a directed graph whose nodes are the atoms of q , and whose edges are formed according to the following rule. There is an edge from an atom a_1 to an atom a_2 if and only if at least one of the following occurs: (i) a_2 depends on a_1 , i.e., if the expression occurring as predicate argument in a_2 is a variable occurring as object argument in a_1 ; (ii) there exists a variable x of Q that is not a metavariable and x occurs as object argument in both a_1 and a_2 (note that in this case, there will be an edge also from a_2 to a_1). Intuitively, a_2 depends on a_1 , if by evaluating the query with body a_1 , one obtains bindings for the metavariable of a_2 , which allow to instantiate and then evaluate the subquery with body a_2 .

2. It assigns to each atom a a *depth* $d(a)$ by using the following strategy:
 - (a) $k \leftarrow 0$;
 - (b) do the following until the graph is empty:
 - 2.1 assign k to each atom n occurring in a strongly connected component C such that every atom in C has all of its incoming edges coming from C ;
 - 2.2 remove from the graph all the considered atoms and their edges;
 - 2.3 $k \leftarrow k + 1$.
3. For every atom a , it marks every variable that is a metavariable and occurs in an atom a' such that $d(a') < d(a)$. Intuitively, if a variable within an atom is marked, then bindings for such a variable will be provided in order to evaluate the query with body that atom by using a standard $DL-Lite_{\mathcal{R}}$ reasoner.
4. For every $i \in \{0, m\}$, where m is the maximal atom depth, it defines an IHCQ q_{γ_i} whose arity is the number of variables occurring in some atom at depth i , and whose form is $q_{\gamma_i}(\underline{\mathbf{u}_{\gamma_i}}, \mathbf{u}_i) \leftarrow b_{\gamma_i}$, where \mathbf{u}_{γ_i} contains all variables that are marked in some atom at depth i , \mathbf{u}_i contains all unmarked variables occurring in some atom at depth i that also belong to x or, for $i < m$ are marked in some atom at depth $i + 1$, and b_{γ_i} is the conjunction of all atoms at depth i . Intuitively, atoms having the same depth will be evaluated in the same query.

The queries $q_{\gamma_1}, \dots, q_{\gamma_m}$ built by $SPLITANDORDER(q)$ will be used by the function DHQPSYNTHESIS to construct the DHQP program whose evaluation will return the answers to q .

Example 5. Consider the IHCQ presented in Example 2. The result of the $SPLITANDORDER$ function is the sequence $B = (q_{\gamma_0}, q_{\gamma_1})$ presented in Example 3. \square

4.3 The Function DHQPSYNTHESIS

Given a query IHCQ q with target tuple x , and the sequence $(q_{\gamma_0}, q_{\gamma_1}, \dots, q_{\gamma_m})$ obtained by $SPLITANDORDER(q)$, DHQPSYNTHESIS builds a DHQP program constituted by $m + 1$ as follows:

- The base rule has the form $I_0(v) \leftarrow \gamma_0(w)$, where each variable of w that is an input variable of q_{γ_0} is underlined, and v is such that: if $m = 0$, then v coincides with x , otherwise v is constituted by all the variables in w that are in x , and all the variables in w in position of input arguments in q_{γ_1} .
- For $j \in \{1, \dots, m\}$, the j -th rule defining I_j based on I_{j-1} has the form $I_j(v) \leftarrow I_{j-1}(u), \gamma_j(w)$ where $\text{def}(\gamma_j) = q_{\gamma_j}$, each variable of w that is an input variable of q_{γ_j} is underlined, and v is such that if $j = m$, then v coincides with x , otherwise v is constituted by all the variables in w that are in x , and all the variables in w in position of input arguments in $q_{\gamma_{j+1}}$.

Example 6. Consider the IHCQ Q presented in Example 2 and the sequence of queries B obtained by executing $SPLITANDORDER(Q)$, as shown in Example 5. It is easy to see that, by executing $DHQPSYNTHESIS(x, B)$, where $x = (x, y, z)$ is the target tuple of Q , we obtain the DHQP program presented in Example 3. \square

4.4 The SBP Algorithm

We are now ready to describe the algorithm SBP. As shown in the following, the algorithm iterates over the various IHCQs in IHUCQ input query Q . At each iteration, the IHCQ q under exam is first split into subqueries, that are ordered on the basis of bindings that the evaluation of a subquery provides for variables of subsequent subqueries. On the basis of the ordering of subqueries, q is then compiled into a DHQP program, and then such a program is evaluated over \mathcal{H} . The result of such evaluation constitutes the resulting set of answers R .

Algorithm $\text{SBP}(Q, \mathcal{H})$

```

input IHUCQ  $Q$  of arity  $n$ ,  $Hi(DL-Lite_{\mathcal{R}})$  KB  $\mathcal{H}$ 
output Set of  $n$ -tuples of expressions in  $\mathcal{E}_{DL-Lite_{\mathcal{R}}}(\mathcal{S})$ 
begin
     $R = \emptyset$ 
    for each  $q \in Q$ 
         $B = \text{SPLITANDORDER}(q);$ 
         $P = \text{DHQPSYNTHESIS}(q, B);$ 
         $R = R \cup \text{EVALUATEDHQP}(P, \mathcal{H})$ 
    return  $R$ 
end

```

The following theorem shows termination and correctness of the algorithm.

Theorem 1. *Given a $Hi(DL-Lite_{\mathcal{R}})$ KB \mathcal{H} and an IHUCQ q over \mathcal{H} , the $\text{SBP}(q, \mathcal{H})$ terminates and computes the certain answers of Q over \mathcal{H} .*

Proof. (Sketch) Termination of SBP follows from termination of its three main functions. As for correctness, it is easy to see that it is sufficient to consider the case of an IHCQ q . Also, to simplify the proof, we assume that q is such that its distinguished variables include all the query metavariables. Indeed, it can be proved that the certain answers to an IHCQ q whose distinguished variables do not include some metavariable can be obtained by computing the certain answers to the query with the same body of q and whose metavariables are all distinguished, and then by appropriately computing a projection of the resulting set of tuples onto the distinguished variables of the initial query. From now on, the proof relies on the correctness of the algorithm COMPUTEPMG, based on the notion of PMG . In particular, let q be an IHCQ and P the DHQP program built from q by SBP. To prove correctness of SBP, we next show that $(*) t \in \text{Evaluate}(P, \mathcal{H})$ iff there exists $q' \in PMG(q, \mathcal{H})$ such that $t \in \text{cert}(q', \mathcal{H})$.

Let us assume to evaluate P and let B be the database consisting of the extensions of the intensional and bridge predicates computed during the evaluation. Clearly, P can be seen as a non recursive Datalog program and, as such, it can be evaluated over B by “unfolding” its rules in order to obtain a single Datalog rule of the form

$$I_m(\mathbf{v}) \leftarrow \gamma_0(\mathbf{u}_0), \dots, \gamma_m(\mathbf{u}_m)$$

that can be evaluated over B . Hence, proving $(*)$ is equivalent to prove that $t \in I_m$ iff there exists $q' \in PMG(q, \mathcal{H})$ such that $t \in \text{cert}(q', \mathcal{H})$.

\Rightarrow : Let $t \in I_m$, x_m be the metavariables of q and t_m be the projection of t onto t_m . Let $q' = INST(q, x_m \leftarrow t_m)$. Clearly, $q' \in PMG(q, \mathcal{H})$. Also, by the standard Datalog semantics, there exists at least one set $W = \{\gamma_0\theta, \dots, \gamma_m\theta\}$ of ‘‘witnesses’’ of t obtained by applying a substitution θ such that $v\theta = t$, that make the conjunction true in B . But then, by construction, and based on the fact that each γ_i collects the certain answers to instantiations of subqueries $\text{def}(\gamma_i)$, $q'_\theta \leftarrow W$ is logically implied by \mathcal{H} . Hence, $t \in cert(q', \mathcal{H})$

\Leftarrow : Let q' be a query in $PMG(q, \mathcal{H})$ and $t \in cert(q', \mathcal{H})$. We next show, by induction on the number m of rules of P that $t \in I_m$. The case of $m = 0$ is trivial, given that $\text{def}(\gamma_0) = q$ and hence I_0 are the certain answers of $PMG(q, \mathcal{H})$. Let us now suppose that $m > 0$. Then, by the semantics of the certain answers to q' , there exists a grounding substitution θ such that $t = v\theta$, where v are the distinguished variables of q' , and the Boolean query $q'_\theta \leftarrow a_1\theta, \dots, a_l\theta$ is logically implied by \mathcal{H} . Let q'_{m-1} be the query obtained from q' by deleting the atoms a_i obtained by instantiating atoms occurring in the body of $\text{def}(\gamma_m)$ as well as by deleting the distinguished variables that occur only in such atoms. Also, let t_{m-1} be the projection of t onto the distinguished variables of q'_{m-1} . Then, $t_{m-1} \in I_{m-1}$ by inductive hypothesis. Now, suppose that the m -th rule has the form $I_m(v) \leftarrow I_{m-1}(u), \gamma_m(w)$. Let t', t'' be obtained by projection of t respectively onto w and onto the variables of w that are underlined. By construction, the Boolean query obtained from q'_θ by retaining only the atoms $a_i\theta$ obtained by instantiating atoms occurring in the body of $\text{def}(\gamma_m)$ is logically implied by \mathcal{H} . Hence t' is a certain answer of $\text{def}(\gamma_m)$ and hence $(t_{m-1}, t') \in I_m$. \square

It is also easy to characterize the complexity of SBP. The following theorem provides such a characterization.

Theorem 2. *Answering IHUCQs over $Hi(DL-Lite_{\mathcal{R}})$ KBs with SBP is PTIME w.r.t. both instance and KB complexity, and NP-complete w.r.t. combined complexity.*

5 Experiments

In this section we show the results of a set of tests that were carried out to compare the performance of SBP with that of the algorithm COMPUTEPMG presented at the end of Section 3. The tests compare the evaluation time of a number of IHCQs expressed over the ontology \mathcal{H} of the Italian Public Debt mentioned in Section 1. The whole ontology \mathcal{H} contains about 100000 assertions of type $Inst_C$ and $Inst_R$, and 412 assertions of type Isa_C , Isa_R , $Disj_C$ and $Disj_R$, giving rise to a set $Concepts(\mathcal{H})$ of cardinality 124. Note that, for computing the certain answers to metaground queries, according to what is required by the function EVALUATEDHQP, we used MASTRO [4].

```

 $q_1(x, y) \leftarrow Inst_C(x, y);$ 
 $q_2(x, y) \leftarrow Inst_C(x, z), Inst_C(z, y);$ 
 $q_3(x, y) \leftarrow Inst_C(x, z), Inst_C(z, w), Inst_C(w, y);$ 
 $q_4(x, y) \leftarrow Inst_C(x, z), Inst_C(z, w), Inst_C(w, v), Inst_C(v, y);$ 
 $q_5(x, y) \leftarrow Inst_C(x, z), Inst_C(z, y), Inst_C(y, \text{Category\_of\_f\_i});$ 
 $q_6(x, y, z, w) \leftarrow Inst_C(x, y), Inst_C(y, w), Inst_C(y, z), Inst_C(w, \text{Category\_wrt\_market}),$ 

```

Table 2. Comparison between SBP and COMPUTEPMG

IHCQ	SBP		COMPUTEPMG	
	# metaground IHCQs	time	# metaground IHCQs	time
q_1	124	0.58 sec.	124	0.56 sec.
q_2	176	2.13 min.	15376	15.12 min.
q_3	205	4.6 min.	1906624	timeout 3 h.
q_4	205	4.9 min.	236421376	timeout 3 h.
q_5	53	16.44 sec.	15376	timeout 3 h.
q_6	25	25.8 sec.	1906624	timeout 3 h.
q_7	85	29.72 min.	15376	timeout 3 h.

$q_7(x, y, z) \leftarrow Inst_R(x, z, duration), Inst_R(y, z, duration), Inst_C(y, w), Inst_C(x, v),$
 $Inst_C(w, Type_of_Italian_f_i), Inst_R(w, t, has_issuer),$
 $Inst_C(t, Public_debt_entity), Inst_C(v, Type_of_foreign_f_i).$

We now briefly describe the set of queries used in the experiments. The IHCQs q_1, q_2, q_3 and q_4 are structural queries, in the sense that they ask for pairs of elements connected through chains of different length of the “instance-of” relation. For example, q_2 asks for all pairs of expressions $\langle e_1, e_2 \rangle$ such that e_1 is an instance of some e' , and e' is an instance of e_2 . The IHCQs q_5, q_6 and q_7 correspond to actual reports the users ask frequently.

- q_5 asks for pairs $\langle x, y \rangle$ such that x is a financial instrument (for example, “ib_03_121”) that is an instance of a type of financial instrument z (for example, “BTP”) such that z is an instance of some instance of “Category of financial instrument type” (for example, “Type of Italian financial instrument”).
- q_6 asks for financial instruments x (for example “jh_09_78”) that are instances of a “Type of financial instrument” y (for example “Commercial paper”), such that y is an instance of both w and z , where w and z are instances of “Category wrt market” and “Category wrt payment mode”, respectively.
- q_7 is the query illustrated in Example 2.

The results of the tests are summarized in Table 2, where each row refers to one of the 7 queries described above. More precisely, each row shows the behaviors of both the algorithm SBP and the algorithm COMPUTEPMG when computing the certain answers to the corresponding query. Such behavior is characterized in terms of both the number of metaground queries sent to the first-order reasoner (in our case, MASTRO), and the time needed to compute the certain answers. It is easy to see that the performances of the two algorithms are comparable only for q_1 . In all other cases, SBP outperforms COMPUTEPMG. By comparing the columns “# metaground IHCQs”, one can realize that the main reason of the improvement is the smarter grounding instantiation of the metavariables of the query performed by SBP with respect to the naive one performed by COMPUTEPMG. Note, however, that the number of metaground queries to be evaluated is not the only factor determining the cost of the algorithm. Indeed, the table shows

that the evaluation time of q_7 is greater than those of q_1, q_2, q_3 and q_4 , even though the number of metaground IHCQs involved is smaller.

Another set of experiments that we carried out (not reported here due to lack of space) aimed to analyze the performances of the two algorithms in the case where we fix one query, and we add intensional assertions (i.e., assertions of type Isa_C , Isa_R , $Disj_C$ and $Disj_R$) over new concept and role expressions. The experiment confirmed that the blind grounding of metavariables performed by COMPUTEPMG causes a deterioration of the performance of such an algorithm. The explanation of this phenomenon could be seen in Table 2: the number of metaground IHCQs considered by COMPUTEPMG when evaluating query q is $|Concepts(\mathcal{H})|^k = 124^k$, where k is the number of distinct metavariables occurring in the body of q .

6 Conclusion

We have presented a new technique to answer IHCUQs over $Hi(DL-Lite_{\mathcal{R}})$ KBs, which improves the one presented in [8]. The improvement is based on the idea of avoiding the blind grounding of metavariables, that often causes query answering to become impractical.

We have illustrated a set of experiments involving real world ontologies showing the advantage of using our strategy: indeed, such experiments confirm that answering IHUCQs with the COMPUTEPMG strategy can be very inefficient, while SBP is a promising direction to pursue.

We are aware that there are other possible optimizations that we can study to improve the work presented here. One notable example is the management of cycles in the dependency graph. Presently, we assign the same depth to all the atoms occurring in a cycle. However, there are cases where breaking the cycle, thus assigning different depth to the involved atoms, can allow to further reduce the number of metaground queries that have to be considered during the evaluation of the corresponding program. Our future works will be focused on identifying those cases, and studying other possible optimization strategies.

References

1. Antonioli, N., Castanò, F., Civili, C., Coletta, S., Grossi, S., Lembo, D., Lenzerini, M., Poggi, A., Savo, D.F., Virardi, E.: Ontology-based data access: the experience at the Italian Department of Treasury. In: Proc. of the Industrial Track of CAiSE 2013. CEUR, vol. 1017, pp. 9–16 (2013), ceur-ws.org
2. Attardi, G., Simi, M.: Consistency and completeness of OMEGA, a logic for knowledge representation. In: Proc. of IJCAI 1981, pp. 504–510 (1981)
3. Badea, L.: Reifying concepts in description logics. In: Proc. of IJCAI 1997, pp. 142–147 (1997)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The Mastro system for ontology-based data access. Semantic Web J. 2(1), 43–53 (2011)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The $DL-Lite$ family. J. of Automated Reasoning 39(3), 385–429 (2007)

6. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: Second-order description logics: Semantics, motivation, and a calculus. In: Proc. of DL 2010. CEUR, vol. 573 (2010), ceur-ws.org
7. Console, M., Lembo, D., Santarelli, V., Savo, D.F.: Graphol: Ontology representation through diagrams (2014)
8. De Giacomo, G., Lenzerini, M., Rosati, R.: Higher-order description logics for domain metamodeling. In: Proc. of AAAI 2011 (2011)
9. Di Pinto, F., De Giacomo, G., Lenzerini, M., Rosati, R.: Ontology-based data access with dynamic TBoxes in DL-Lite. In: Proc. of AAAI 2012 (2012)
10. Glimm, B., Rudolph, S., Volker, J.: Integrated metamodeling and diagnosis in owl 2. In: Proc. of ISWC 2010, pp. 257–272 (2010)
11. Lehmann, F. (ed.): Semantic Networks in Artificial Intelligence. Pergamon Press, Oxford, United Kingdom (1992)
12. Motik, B.: On the properties of metamodeling in OWL. J. of Logic and Computation 17(4), 617–637 (2007)
13. Mylopoulos, J., Bernstein, P.A., Wong, H.K.T.: A language facility for designing database-intensive applications. ACM Trans. on Database Systems 5(2), 185–207 (1980)
14. Pan, J.Z., Horrocks, I.: OWL FA: a metamodeling extension of OWL DL. In: Proc. of WWW 2006, pp. 1065–1066 (2006)

Knowledge-Based Compliance Checking of Business Processes

Ildikó Szabó and Krisztián Varga

Department of Information Systems, Corvinus University of Budapest
1093 Budapest, Fővám tér 13-15., Hungary
iszabo@informatika.uni-corvinus.hu,
kvarga@informatika.uni-corvinus.hu

Abstract. Discovering discrepancies between actual business processes and business processes outlined by standards, best practices facilitates decision making of process owners regarding business process improvement. In semantic compliance management, knowledge related to job roles can serve as a basis to check compliance. Process ontologies preserve the structure of business processes and bear background knowledge for executing processes, which can be structured in domain ontologies. This paper presents how process and domain ontologies can work together in ontology matching to make a proposal for process owners regarding compliance check. A case from nursing practice will illustrate the applicability of this approach.

Keywords: compliance checking, process ontology, domain ontology, business process, ontology matching.

1 Introduction

Laws, regulations, standards, best practice can serve as points of reference in business process compliance checking because compliance requirements are folded in them. But these documents per se hide knowledge about business processes regularized by them and put these processes into context. If this knowledge is structured into ontologies, it facilitates compliance checking of actual business process with reference business processes extracted from guidelines in semantic manner.

This paper presents a compliance checking process which contains the next steps. BPM tool is designated to create business process models based on actual processes or from guidelines as reference documents. XSLT transformation creates process ontologies from these models. Ontology learning tool is responsible for extracting knowledge elements from these models and identifying them in domain ontologies. The domain ontologies are stored in an ontology driven learning environment called Studio which is continually developing in European-funded projects. Ontology matching tool built-in Protégé 4.X ontology development environment serves as the inference engine to compare the actual business process with the reference business process working on the same domain.

The goal of this process to discover the discrepancies between actual business process and reference business process working on the same domain at process level and task level too. We search for answers to questions, like how the actual process is overlapped by the reference process, regarding the common and different tasks, I/O elements, roles, IT resources? How the roles in actual process are overlapped by the roles in reference process regarding knowledge required by task which is performed by a given role? The second question builds up a layered query, since at first, it investigates the tasks, than the whole process, than the whole process group.

The process owners can gain information about the task distribution of the processes, the overlapping of these processes and process elements, in the light of the background knowledge stored in domain ontologies. According to this report, the business process can be reorganized to meet the requirements given by references.

This stage of the research metrics aren't used to measure the goodness of actual processes. Instead of it, it emphasizes the role of domain ontologies for various reasons. In an organization, job roles are sets of tasks, which can be organized to business processes. Job roles are filled by employees, who need to have a required level and set of competences and knowledge to perform their tasks. So the elements of this triple (role-task-knowledge) aren't independent from each other [1]. Information systems provide knowledge for running processes. Hence business processes lean on a specific knowledge asset that may be a bottleneck in running processes. The process can be stopped if the required knowledge is missing. Nevertheless domain ontology can help to identify the semantically common process elements in both process ontologies. Hence they can be interpreted by a standard format in order to matching them.

The paper is structured as follows. Our approach will be placed among semantic compliance checking researches in Section 2. The role of ontology-based methods will be presented in Section 3. The full compliance checking process will be illustrated a case from nursing practice in Section 4.

2 Semantic Compliance Checking

Compliance checking is an important area in semantic business process management. Combining the semantics with the business process management provides an opportunity not only to model processes but define intelligent queries on them, and check their compliance with regulations, laws, best practices or other standard processes [2].

For compliance checking, related work can be found in the area of internal controls and risk, such as [3] and [4] where an approach for semantic compliance management for BPM is presented. In [4] the authors integrate compliance requirements to processes in the phase of business process design.

In [5], another approach for business process-based compliance management is presented. It defines an extension to a business process meta-model for regulatory compliance. However, the approach does not incorporate ontologies and thus, does not profit from the power of semantic technologies.

An ontology-based approach is described in [6], where a compliance ontology is designed and proposed to be integrated in business process models.

Ly et al. [7] presents a framework for semantic process verification to ensure system correctness after arbitrary process changes by defining semantic constraints over processes. El Kharbili et al. [8] presents eight requirements for a compliance management framework and discuss different ways of conducting compliance checking with a proposed policy-based framework for business process compliance management. They emphasize the use of business process ontologies as well.

Semantic framework for compliance management in business process management was elaborated in SUPER project. The architecture of this framework starts from the formalization of regulations into semantic policies, business rules. It uses Policy Ontology and Business Rules Ontology for modeling business rules to extend the modeling for business process. Business processes cannot be independent from their contexts, hence these ontologies uses the related business vocabulary structured into a domain ontology [9] [10].

Our approach aligns to this framework in that sense it formalizes compliance requirements, uses domain ontologies for grabbing the context and runs an inference engine for getting report about the compliance. But the reference business processes interpret the compliance requirements and the background knowledge required to run business processes gets more emphasis.

3 Ontology-Based Approach

Ontologies facilitate to create a process model by providing a unified view about the elements, their attributes and their relationships of this model. The process ontology is capable of representing a real business process. We can build them by reusing or extending an existing ontology (like PSL Ontology [5]), using a framework (like the framework of SUPER project [11]) or transforming the output of a BPM tool (like ARIS or ADONIS) into an ontology format [12].

The importance of process ontologies in SBPM is to help to exchange process information between applications in the most correct and complete manner [9], or to restructure business processes by providing a tool for examining the matching of process ontologies [13]. Semantic or structural discrepancies between process ontologies may be derived from different viewpoint or background knowledge used by the experts in the process of building ontology, or the different structure of real business processes. An ontology matching procedure applied in the domain of process ontologies helps to reveal these discrepancies in order to prepare the decision making about business process reengineering.

In addition to application possibilities in industry, the usability of ontologies in bridging of semantic differences for administrative processes was exemplified [14].

The objective of ontology learning is “to generate domain ontologies from various kinds of resources by applying natural language processing and machine learning techniques” [15]. We can distinguish statistical, rule-based or hybrid ontology learning technique. The pattern-based ontology learning technique is one of them [16].

Ontology learning is an appropriate method to extract knowledge from process ontologies and use them to broaden domain ontologies with new elements. Meanwhile

the extended version of this tool can connect process ontologies with domain ontologies through these knowledge elements.

Alasoud et al. [17] define ontology matching problem as follows: “given ontologies O₁ and O₂, each describing a collection of discrete entities such as classes, properties, individuals, etc., we want to identify semantic correspondences between the components of these entities.”

The goals of combining ontologies are to merge, transform, integrate, translate, align or map etc. them into a new or an existing ontology. The general ontology mapping tools enumerated by Noy [18] or Choi et al. [19], or developed by Protégé community use different kind of method to identify the semantic correspondences between two ontologies. So we can distinguish methods searching axiomatic correspondences (e.g. OWLDIFF [20], Compare Ontologies function in Protégé 4.X) or calculating similarity values etc... The latters take probability distributions (e.g. Glue [21], OMEN [22]) or text similarity functions (e.g. LOM [23]) as a basis.

Process specific methods were elaborated by Jung [13] used logical assertions and similarity measures to facilitate the interoperability among processes. Koschmidier and Oberweis [24] used Petri nets „to obey an operational semantics that facilitates composition, simulation, and validation of business processes“.

4 Use Case: Ontology Based Compliance Checking of the Post-operative Nursing Care Process

We demonstrate our approach of semantic business process matching using the methods from project ProKEx¹, with the “Post-operative nursing care” process investigated in the Med-Assess² project.

4.1 Overview of Project ProKEx

ProKEx aims to deliver an integrated platform to support corporate knowledge management by semantic business process management. The project will provide a solution to extract, organize and preserve knowledge embedded in organizational processes to enrich organizational knowledge base in a systematic and controlled way, support employees to easily acquire their job role specific knowledge and help to govern human capital investment.

The ProKEx method to extract and transfer knowledge elements from business process models gives us the opportunity to enrich the semantic business process management’s compliance check with knowledge-based aspects, and investigate the post-operative nursing care process relied not only the name and sequence of process tasks but the knowledge required for tasks and job roles, and the knowledge required to the whole process.

The architecture of ProKEX system was presented by Arru [35].

¹ ProKEx: Integrated Platform for Process-based Knowledge Extraction, EUREKA project.

² Med-Assess: Adaptive Medical Profession Assessor, Leonardo da Vinci project.

4.2 Overview of Project Med-Assess

Project Med-Asses is an innovation transfer project in the context of the LEONARDO DA VINCI innovation transfer call as a part of the EU-Programme of Lifelong Learning. The previous developed solution of OntoHR³ will be transferred from the IT domain to the health domain in a multi-lingual environment.

Med-Assess provides a solution where nurses may assess their knowledge and skills to get an insight in which areas they might need refreshment or further training. Additionally Med-Assess will support superiors in a hospital or other medical institution with recruitment tests for new employees, in the measuring of knowledge, abilities and competencies of current employees as well as providing learning content or material for VET (Vocational Education Training) on the job. With regard to the international context, Med-Assess can be used for determining whether a foreign job applicant holds similar knowledge and qualifications than a local applicant, or if he will need additional training to fulfill the job related tasks.

In the project we have modeled the business processes related to the nurses in an institution which is an international private clinic specialized on diagnosis and therapy of head and spinal diseases.

4.3 The Post-operative Nursing Care Process

Post-operative nurses provide intensive care to patients as they awaken them from anesthesia after a surgical procedure. Because they typically have significant experience in a medical-surgical environment or in emergency medicine, they are equipped to identify complications and intervene quickly. As this process is carried out on a daily basis, it may seem to be a routine set of activities, but it requires a lot of knowledge.

Regular observations made in the postoperative period assist the health-care team to build up a complete picture of the patient's condition following surgery. The findings should be compared against baseline observations taken pre-operatively and against previous postoperative readings, as these will assist nurses accurately to chart the patient's progress [25].

There are two common methods of monitoring patients following surgery, clinical monitoring and general observation of the patient [26]. Postoperative care begins in the recovery room and continues throughout the recovery period. Critical concerns are airway clearance, pain control, mental status, and wound healing. Other important concerns are preventing urinary retention, constipation, deep venous thrombosis, and BP variability (high or low).

In the project Med-Assess, we have modeled the process "Post-operative nursing care" by making interviews with nurses from a specific institution. They have told us what they do, why do they do it, and how do they make their decisions when they monitor patients. We have transformed these interviews into business processes, modelled them with BOC Group's Adonis Business Process Modeling Tool, which is

³ OntHR: Ontology Based Competency Matching, Lifelong Learning Programme project.

a graphical business process management tool [27], and populated the process tasks' description (which is a basic text field) with the relevant data from the interviews.

Based on the interviews during the post-operative nursing care nurses have to check the patients' blood pressure, their body temperature and heart frequency, they have to care the wounds, ask for the pain and make any other kinds of check they think are necessary. The nurses have to decide whether any other specific test is needed, and they have to prepare the patient for that, and sometimes they even have to do these lab tests. The nurses have to evaluate all of the findings, and decide whether a doctor is needed urgently, or they can intervene without a doctor. After that, they have to document everything.

This process was investigated by us in the institution, but the question is that whether it is "good" or not? Does it meet the standards and best practices, or not? Semantic business process management's compliance checking of business processes may give the answer to the question.

We have not found a "one and only" standard for post-operative nursing care, but we did find several best practices, learning materials and articles about the topic. As we see, this is a general issue in many areas; therefore we try to create a framework that can handle the exploration of standards in the internet. The idea of open data and linked open data underlines the importance of this aspect and justifies our approach.

The reference post-operative nursing care process consists of the following: the nurse has to check blood pressure, body temperature and heart frequency, he or she has to take care of the wounds, manage the pain and the respiration and airways, the nurse has to monitor fluid level and GI function, and has to check the level of consciousness. After these checks, the nurse has to compare the results with the baseline values, do the necessary actions and document everything.

4.4 Process Ontology

In our proposed approach, we discuss how to establish the links between model elements and ontology concepts. Ontologies basically provide semantics and they can describe both semantics of the modeling language constructs as well as semantics of model instances. [28] There were several projects investigating business process ontologies such as the SUPER⁴ project, but in our solution we are using a process ontology we have created in our own way.

For the extension and mapping the conceptual models to ontology models, the models are exported in the structure of ADONIS XML format. All objects from the business process model will be an 'instance' in the XML structure, the attributes have the tag 'attribute', while the connected objects (such as the performer, or the input/output data, which are stored in another model in the Adonis tool) have the tag 'interref'.

Our model transformation approach aims at preserving the semantics of the business model. The general rule we follow is to express each ADONIS model element as a class in the ontology and its corresponding attributes as attributes of the class. This

⁴ <http://ip-super.org>

transformation is done with an XSLT script that performs the conversion. The converted OWL ontology in the structure of Protege/OWL XML format is imported into the editor of Protege 4.X. A sample part of the transforming XSLT code (mapping the ‘Responsible role to an ontology element) can be seen in Figure 1.

```

<xsl:if test="INTERREF[@name='Responsible role']/IREF">
<ObjectAllValuesFrom>
    <ObjectProperty IRI="#performed_by"/>
    <xsl:for-each select="INTERREF[@name='Responsible role']/IREF">
        <Class>
            ....
            <xsl:attribute name="IRI">#<xsl:value-of select=
                "functx:words-to-camel-case(@tobjname)" /></xsl:attribute>
        </Class>
    </xsl:for-each>
</ObjectAllValuesFrom>
</xsl:if>
```

Fig. 1. Fraction of XSLT code transforming ‘Responsible role’ attribute to an ontology element

To specify the semantics of ADONIS model elements through relations to the ontology concepts, the ADONIS business model first must be represented within the ontology. In regard to the representation of the business model in the ontology, one can differentiate between a representation of ADONIS model language constructs and a representation of ADONIS model elements. ADONIS model language constructs such as “activity”, as well as the control flow are created in the ontology as classes and properties. Subsequently, the ADONIS model elements can be represented through the instantiation of these classes and properties in the ontology. The linkage of the ontology and the ADONIS model element instances is accomplished by the usage of properties. These properties specify the semantics of an ADONIS model element through a relation to an ontology instance with formal semantics defined by the ontology. Figure 2. shows a fragment of the process ontology, where the “Ask for the pain” task, its annotation and responsible role can be seen.

```

<SubClassOf>
    <Annotation>
        <AnnotationProperty abbreviatedIRI="dc:description"/>
        <Literal datatypeIRI="&xsd:string">Check pain level by asking the patient.</Literal>
    </Annotation>
    <Annotation>
        <AnnotationProperty abbreviatedIRI="&xdfs:label"/>
        <Literal datatypeIRI="&xsd:string">Ask for the pain</Literal>
    </Annotation>
    <Class IRI="#AskForThePain"/>
    <ObjectAllValuesFrom>
        <ObjectProperty IRI="#performed_by"/>
        <Class IRI="#Nurse"/>
    </ObjectAllValuesFrom>
</SubClassOf>
```

Fig. 2. Fraction of the business process ontology

Since we want to use the process model not only as a structural definition of tasks, but as the holder of the required knowledge to each task and their responsible roles, we have run text-mining algorithms to gather knowledge elements from the process models. The output of the text-mining was an aim when merging the nursing domain ontology (see next section) with the process ontology.

4.5 Nursing Ontology in the Studio System

Studio is a competence-based e-learning methodology and system which “provides support in exploring missing knowledge areas of users in the frames of an ontology driven e-learning environment in order to help them to complement their educational deficiencies” [29]. Educational Ontology is the core of this system. It maps knowledge into <Knowledge area>, <Basic concept>, <Theorem> and <Example> classes. It reflects the hierarchy of knowledge areas by using relations called “has_part” and “has_subknowledge_area”. It orders the knowledge elements into a cognitive sequence too by using relations called “required_knowledge_of”, “premise”, “conclusion” and “refers_to”. The detailed description was presented by Vas et al. [30].

The Studio system has been already used in several projects [31], hence several domain ontologies was merged into it, using Education Ontology as meta-model. Nursing ontology was created by reason of the MedAssess project (see Fig. 3).

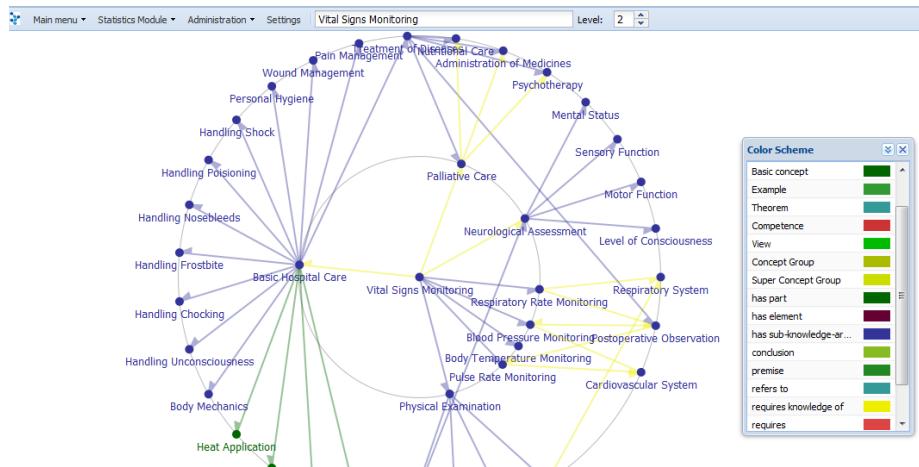


Fig. 3. Nursing ontology in the Studio system

4.6 Merging Process Ontology with Domain Ontology

The Nursing ontology was exported into RDF format from the Studio system, and imported into Protégé 4.X. The development of ontology learning tool is under development under the aegis of ProKEx project. The extraction of knowledge elements from the business process models has already been elaborated [33], but the method to

identify them in the Studio system hasn't been developed yet. Hence having merged the process ontologies with the Nursing ontology by using the feature called Merge ontologies of Protégé 4.X and set the connections between the tasks and their knowledge elements manually, the next ontology was created (Fig. 4).

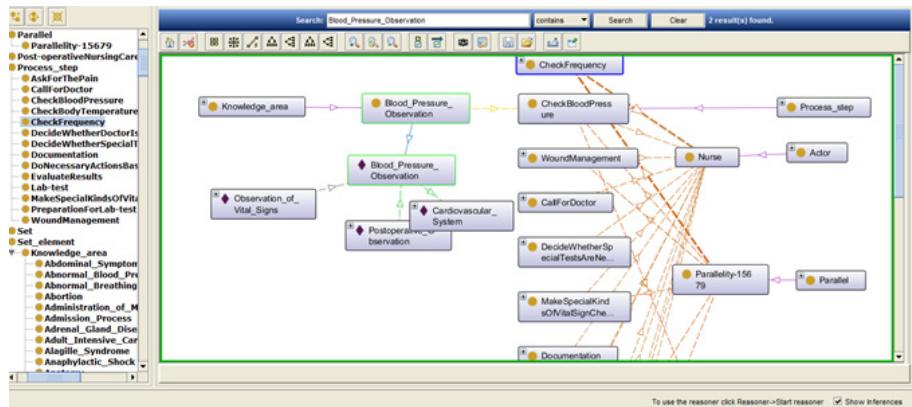


Fig. 4. Merged ontologies related to the actual process in Protégé 4.X

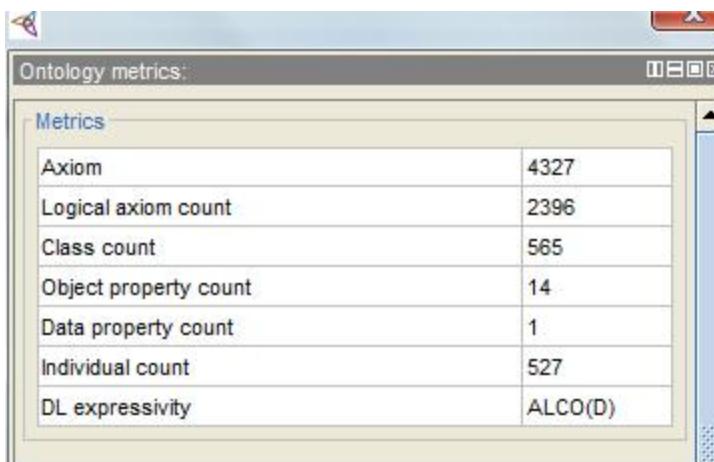


Fig. 5. Statistics of the ontology

4.7 Domain Ontology Based Compliance Checking

Having created the appropriate base to start executing the domain ontology based compliance checking, we investigated the role of this merged ontology in the next compliance checking questions:

1. How the actual process is overlapped by the reference process, regarding the common and different tasks, I/O elements, roles, IT resources?

In some cases, actual process contains concretes (e.g. Omron® 7) instead of its related general concept (e.g. blood pressure monitor). Hence this question outlines structural investigation instead of semantic investigation, the domain ontology and other semantic techniques (e.g. similarity calculation between process elements) can facilitate this task. A solution using ontology matching was presented by Ternai et al. [32].

2. How the roles in actual process are overlapped by the roles in reference process regarding knowledge required by task which is performed by a given role?

- (a) Is the required knowledge the same from task to task in the actual and the reference process?
- (b) Is the required knowledge the same for the same job roles in the whole actual and the reference process?
- (c) Is the required knowledge the same for the same job roles in the actual and the reference process group?

These questions build up a layered query, since at first, it investigates the tasks, than the whole process, than the whole process group. In this way, the process owner can have a detailed report based on the knowledge required to execute processes.

The report can show to the process owner:

Tasks

- (a) In the actual process, which knowledge elements has been identified;
- (b) Which of them are part of the reference process;
- (c) Which of them are not part of the reference process.

Processes

- (a) In the actual process, which knowledge elements has been identified;
- (b) Which of them are part of the reference process;
- (c) Which of them are not parts of the reference process.

Job roles

- (a) In the actual process, which knowledge elements has been identified;
- (b) Which of them are part of the reference process;
- (c) Which of them are not part of the reference process.

As we have seen, role-task-knowledge triple is inseparable (in Section 1), hence these knowledge-specific questions can be supported by the merged ontology more than the first question. The next practical work is about that one.

4.8 A Tool for Semantic Compliance Checking

Protégé 4.X ontology development environment served as a basis to develop a tool which can manage semantic compliance checking in the sense of our approach. It is a

free, Java-based, “open source ontology editor and framework for building intelligent systems” [34].

To elaborate an automatic tool that compares knowledge-specific parts of both processes with each other, we needed to focus on these parts during the matching. We used next DL queries to determine these parts:

Table 1. DL Queries

Description	DL Query
knowledge possessed by a given role	required_by some (performed_by only <Role>)
knowledge required to execute the process	required_by some (belongs_to only <Process>)
knowledge required by a given task	required_by some (<Process_step>)

Having selected the relevant sub-ontologies by using the appropriate DL Query, the elements of these sub-ontologies was differentiated by adding the data property called “type”. This data property had value “actual” or “ref” within actual or reference ontology. The Compare Ontologies built-in function in Protégé 4.X was executed to collect the common and different knowledge element mentioned in the previous section. The main goal of Compare Ontologies function is to compare two versions of the same ontology, so it requires an original and an updated version of the ontology. In our case, the ontology model related to the actual process used in nowadays was considered as the original model. The updated version was the ontology model related to the reference process. Its report is split into three blocks:

Created blocks contain process elements that are in reference model and not in actual model.

Deleted blocks contain process elements that are in actual model and not in reference model.

Modified or Rename and modified blocks contain process elements that are in both ontologies, but they may have axiomatic differences (e.g. in properties or relations).

The elements of Nursing ontology were in Modified blocks, because the same Nursing ontology was connected to “actual” process ontology and “reference” process ontology too. But the knowledge elements from “actual” process ontology had “actual” data property, the ones from “reference” process ontology had “ref” data property.

The technical report created by this inference engine was interpreted by an algorithm to create the next report. The algorithm uses OWL API functions to count the number of tasks and extracts other information about the ontology classes (task, role etc.). It identifies the knowledge elements appeared the above-mentioned blocks (Created, Deleted etc. blocks). The presence of the knowledge elements in these blocks determines their place in the report (e.g. knowledge elements from Created blocks appear in the section called Knowledge that reference role possess, but your role doesn’t).

```

1 ...
2 -----
3 Created RespirationAndAirwayManagement
4 -----
5 Added: Class: RespirationAndAirwayManagement
6 Added: RespirationAndAirwayManagement SubClassOf Process_step
7 Added: RespirationAndAirwayManagement SubClassOf followed_by only Merging-15682
8 Added: RespirationAndAirwayManagement SubClassOf performed_by only Nurse
9 -----
10 ...
11 -----
12 Deleted AskForThePain
13 -----
14 Deleted: Class: AskForThePain
15 Deleted: AskForThePain SubClassOf Process_step
16 Deleted: AskForThePain SubClassOf followed_by only Merging-15682
17 Deleted: AskForThePain SubClassOf performed_by only Nurse
18 -----
19 Modified Body_Temperature_Observation -> Body_Temperature_Observation
20 -----
21 Added: Body_Temperature_Observation type "ref"^^string
22 Deleted: Body_Temperature_Observation type| "actual"^^string

```

Fig. 6. Technical report created by ontology matching tool

***** Match By Id *****
Algorithm Match By Id completed.
Match By Id Step took 172 ms.
1100 owl:entities matched up.
4200 associations matched up.

Took 328ms
Calculating presentation
 Took 1794ms
Total time = 1794ms
Semantic Compliance Checking Report
Process name: Post operative nursing care process
Number of tasks: 14
Role: Nurse
Date: 15-7-2014 12:56
Knowledge that reference role possess, but your role doesn't
abnormal breathing
alveoli
apnea
asthma
bronchi
bronchial disease
bronchectasis
bronchitis
chest pain
chest symptom
chronic cough
cough
dry cough
dyspnea
dyspnoea
hemoptysis
interstitial lung disease
lower respiratory tract
lower respiratory tract disease
lung
lung disease
lung infection
observation of vital signs
Knowledge that reference role doesn't require, but your role possess
Knowledge that reference role and your role possess
abnormal blood pressure
anemia
angiomyolysis
angiofibroplasia
aortic disease
arrhythmia
arterial occlusive disease
arteriosclerosis
bacteremia
blood pressure observation
blood vessel
bloodstream symptom
body temperature observation
bradycardia

Fig. 7. Compliance checking report

It gives a feedback for process owner about that the process model of actual process is more detailed than the reference one, regarding the number of tasks. But the Nurse role has missing knowledge. The interpretation of this report is always the duty of the process owner. In a small organization, where the separation of job roles aren't that wide as in a big company, the report may show that more knowledge is related to one role than the reference recommends. But it doesn't mean, that the actual processes and the way the small company operates is "bad".

Our approach gives the process owner a tool, to investigate not only the processes as a structure, but as a set of knowledge-related activities. In this way, managing business processes and their compliance to the best practices or legalizations will be more sophisticated, it will reflect to the knowledge hiding in the processes.

This tool was written in Java, using OWL API, DLQueryExample.java and SVN repository of Compare Ontologies function.

5 Conclusion and Future Work

Knowledge is required to run business process and determine their context. This paper presented a compliance checking method using ontology-based technique which leaned on domain ontologies and process ontologies. In our research, actual and reference process model are compared in the viewpoint from their knowledge assets. Process ontologies created from process models used XSLT transformation are merged with domain ontologies exported from Studio system. For setting their connection an ontology learning tool is developing. Ontology matching tool is served to compare these ontologies and create report for process owners. This paper presented this solution by an example. This tool can answer the questions mentioned in Section 4.8. It can give feedback at process level, task level and role level that depends on which DL Query is used to create the appropriate sub-ontology.

In the future, we will test this solution regarding other questions asked by process owners. We will investigate how domain ontology can help to calculate the similarity of actual and reference process models. Next to the above-mentioned ontology learning tool, we want to elaborate a tool that creates process ontology from legislations, standards, best practices etc. We want to develop our XSLT script as well, to handle process groups. We will test our approach with more domains, IT operations with ITIL, IT audit and security with COBIT and ISO27001.

Acknowledgement. “This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.”

References

1. Woodruffe, C.: What is meant by a competency? *Leadership and Organization Development Journal* 14(1), 29–36 (1993)
2. Hepp, M., Roman, D.: An Ontology Framework for Semantic Business Process Management. In: Proceedings of Wirtschaftsinformatik 2007, Karlsruhe, February 28–March 2 (2007)
3. Namiri, K., Stojanovic., N.: A Formal Approach for Internal Controls Compliance in Business Processes. In: 8th Workshop on Business Process Modeling, Development, and Support (BPMDS 2007), Trondheim, Norway (2007)
4. Sadiq, W., 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)

5. Karagiannis, D.: A Business process Based Modelling Extension for Regulatory Compliance. In: Proceedings of the Multikonferenz Wirtschaftsinformatik 2008, Munich (2008)
6. Schmidt, R., Bartsch, C., Oberhauser, R.: Ontology-based representation of compliance requirements for service processes. In: Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (2007)
7. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. *Data & Knowledge Engineering* 64, 3–23 (2007)
8. El Kharbili, M., Stein, S., Markovic, I., Pulvermüller, E.: Towards a Framework for Semantic Business Process Compliance Management. In: Sadiq, S., Indulska, M., Zur Muehlen, M. (eds.) Proceedings of the workshop on Governance, Risk and Compliance for Information Systems (GRCIS 2008), vol. 339, pp. 1–15. CEUR Workshop Proceedings, Montpellier (2008)
9. Staab, S., Studer, R.: *Handbook on ontologies*. Springer (2009)
10. El Kharbili, M., Pulvermüller, E.: A Semantic Framework for Compliance Management in Business Process Management. In: BPSC, pp. 60–80 (2009)
11. El Kharbili, M., et al.: Towards a framework for semantic business process compliance management. In: Proceedings of the Workshop on Governance, Risk and Compliance for Information Systems, pp. 1–15 (2008)
12. Kő, A., Ternai, K.: A Development Method for Ontology Based Business Processes. In: eChallenges e-2011 Conference Proceedings, IIMC International Information Management Corporation Ltd, Florence (2011)
13. Jung, J.J.: Semantic business process integration based on ontology alignment. *Expert Systems with Applications* 36(8), 11013–11020 (2009)
14. Herborn, T., Wimmer, M.A.: Process Ontologies Facilitating Interoperability in eGovernment A Methodological Framework. In: Hinkelmann, K., Karagiannis, D., Stojanovic, N., Wagner, G. (eds.) Proceeding of the Workshop on Semantics for Business Process Management at the 3rd European Semantic Web Conference 2006, Budva, Montenegro, pp. 76–89 (June 2006)
15. Haase, P., Völker, J.: Ontology Learning and Reasoning — Dealing with Uncertainty and Inconsistency. In: da Costa, P.C.G., d'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) URSW 2005 - 2007. LNCS (LNAI), vol. 5327, pp. 366–384. Springer, Heidelberg (2008)
16. Zhou, L.: Ontology learning: state of the art and open issues. *Information Technology and Management* 8, 241–252 (2007)
17. Alasoud, A. et al.: An Effective Ontology Matching Technique. In: An, A. et al. (eds.) Foundations of Intelligent Systems. pp. 585–590 Springer Berlin / Heidelberg (2008).
18. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.* 33, 65–70 (2004)
19. Choi, N., et al.: A survey on ontology mapping. *SIGMOD Rec.* 35, 34–41 (2006)
20. OWLDiff: OWL Diff Documentation (2008), <http://krizik.felk.cvut.cz/km/owldiff/documentation.html>
21. Doan, A.H., et al.: Ontology matching: A machine learning approach. In: Staab, S., Studer, R. (eds.) *Handbook on Ontologies*, pp. 385–516 (2004)
22. Mitra, P., Noy, N., Jaiswal, A.R.: OMEN: A Probabilistic Ontology Mapping Tool. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 537–547. Springer, Heidelberg (2005)
23. Li, J.: LOM: a lexicon-based ontology mapping tool. *Information Interpretation and Integration Conference, I3CON* (2004)
24. Koschmider, A., Oberweis, A.: Ontology based business process description. In: Proceedings of the CAiSE, pp. 321–333 (2005)

25. Royle, J., Walsh, M.: Watson's Medical Surgical Nursing and Related Physiology. Baillière Tindall, London (1992)
26. Alexander, M. Fawcett, J., Runciman, P.: Nursing Practice: Hospital and home. The adult. Churchill Livingstone, London (1994)
27. BOC Group: Adonis Product details (2014), <http://www.boc-group.com/products/adonis/product-details/>
28. Kramler, G., Murzek, M.: Business Process Model Transformation Issues (2006)
29. Studio – Ontology Driven Learning Environment,
<http://corvinno.hu/web.nsf/do?open&lang=en&page=proj-studio#>
30. Vas, R., Kovacs, B., Kismihok, G.: Ontology-based mobile learning and knowledge testing. International Journal of Mobile Learning and Organisation 3, 128 (2009)
31. Innovation projects,
<http://corvinno.hu/web.nsf/do?open&lang=en&page=innovacio>
32. Ternai, K., Szabó, I., Varga, K.: Ontology-based compliance checking on higher education processes. In: Kö, A., Leitner, C., Leitold, H., Prosser, A. (eds.) EDEM 2013 and EGOVIS 2013. LNCS, vol. 8061, pp. 58–71. Springer, Heidelberg (2013)
33. Gillani, S.A., Kö, A.: Process-based knowledge extraction in a public authority: A text mining approach. In: Kö, A., Francesconi, E. (eds.) EGOVIS 2014. LNCS, vol. 8650, pp. 91–103. Springer, Heidelberg (2014)
34. Protégé, <http://protege.stanford.edu/>
35. Arru, M.: Application of Process Ontology to Improve the Funding Allocation Process at the European Institute of Innovation and Technology. In: Kö, A., Francesconi, E. (eds.) Electronic Government and the Information Systems Perspective, pp. 133–147. Springer International Publishing (2014)

Improved Automatic Maturity Assessment of Wikipedia Medical Articles^{*} (Short Paper)

Emanuel Marzini, Angelo Spognardi, Ilaria Matteucci, Paolo Mori,
Marinella Petrocchi, and Riccardo Conti

IIT-CNR, Pisa, Italy
`firstname.lastname@iit.cnr.it`

Abstract. The Internet is naturally a simple and immediate mean to retrieve information. However, not everything one can find is equally accurate and reliable. In this paper, we continue our line of research towards effective techniques for assessing the quality of online content. Focusing on the Wikipedia Medicinal Portal, in a previous work we implemented an automatic technique to assess the quality of each article and we compared our results to the classification of the articles given by the portal itself, obtaining quite different outcomes. Here, we present a lightweight instantiation of our methodology that reduces both redundant features and those not mentioned by the WikiProject guidelines. What we obtain is a fine-grained assessment and a better discrimination of the articles' quality, w.r.t. previous work. Our proposal could help to automatically evaluate the maturity of Wikipedia medical articles in an efficient way.

1 Introduction

Recent studies report that Internet users are growingly looking for health information through the Web, by either consulting search engines, social networks, and specialized health portals: in 2013 [14], “one in three American adults have gone online to figure out a medical condition”. Further, in 2013 almost one million US families used video consultations with physicians, mainly through dedicated web portals [12]. The quest for medical information is eased by a myriad of Internet websites with health-related hypertexts. For example, the Wikipedia Medicine Portal is a collaboratively edited multitude of articles concernin health whose consultation spans from patients to healthcare professionals [13,7]. According to a report on online engagement by IMS Health (a world’s leading company dedicated to healthcare), 50% of surveyed physicians who use the Internet have consulted Wikipedia for medical information [1].

However, finding reliable medical articles is an important issue that is worth addressing. For instance, [14] reports that, on the totality of people that searched for health answers on the Web, only 41% say “a medical professional confirmed

^{*} Work partially supported by the Tuscany region projects MyChoice and PICs and by the Registro.it project My Information Bubble.

their diagnosis". Both government departments and scientific reports have raised reliability issues of online seeking health information, see, *e.g.*, [16,19].

This paper proposes an automatic approach for the evaluation of online articles for assessing their quality. Our data-set is the entire collection of articles published on the Wikipedia Medicine Portal. In [4], we proposed a newly-defined metric, the *maturity degree*, to evaluate the quality and reliability of each article. The maturity degree was calculated adopting the Analytic Hierarchy Process (AHP) [15], a well-known methodology for multi-criteria decision making. We showed that the maturity degree is a different metric with respect to the quality level attached to articles by the WikiProject quality assessment. We concluded that a gap exists between the quantitative features that can be computed as metadata of an article and the qualitative features exploited by the quality assessment process of the portal. However, in order to use automatic techniques for article evaluation (like the approach shown in [4]), making use of only quantitative features would greatly ease the process.

Starting from these premises, in this paper we contribute by *i*) pruning the list of features considered for the automatic evaluation of the article (w.r.t. [4]) and experimentally proving that the deleted information do not lead to a fine quality evaluation; *ii*) exploiting the *cosine-similarity* to compare the results obtained with the restricted set of features with respect to the results with the whole set of features. We find out that, besides being more efficient, the new approach also achieves better results in evaluating the maturity of the articles w.r.t. our previous instantiation.

The paper is organized as follows: next section discusses related work. Section 3 briefly recalls the notion of maturity degree and presents our new results, comparing them with previous ones. Section 4 concludes the paper.

2 Related Work

A series of recent work focuses on the assessment of Wikipedia articles, testifying the quest for effective and efficient techniques supporting the community to identify the best-quality material. WikiProject itself has listed a set of criteria to be manually evaluated, useful to determine the quality of an article. In most cases, such criteria express qualitative properties more than quantitative ones, like, for example, comprehensiveness and neutrality. Undoubtedly, such properties are of particular relevance for the evaluation of an article. However, considering them could complicate the process of automatizing the assessment. Instead, this paper focuses on articles features that can be automatically extracted and processed in order to globally evaluate the articles quality level.

Work in [17,2,3,6,23,21] mainly concerns the recognition of featured articles (FA) (articles representing, according to WikiProject, excellent contributions). Recognition is based on features as the number of times an article has been edited (edit time), the word count, and the number of editors. Similarly, [18] and [9] evaluate the articles exploiting the relation between editor and text quality.

Work in [20,22] relies on a different approach to assign an article to one of the existing WikiProject classes. In [22], the authors consider 28 criteria,

grouped into four macro-criteria: lingual, structural, historical, and reputational. They use seven different neural networks for the classification of each article. Overall, each criterion is differently weighted according to the considered class, *e.g.*, linguistic criteria are more important than others to recognize articles in the lowest classes, while richness of content and articulated structure are important for articles of the highest classes. In [20], the authors use also the Wikipedia template messages (small notes to inform readers and editors of specific problems within articles or sections) as new features to assess the quality of the articles.

In line with the results of [21,2,17], which focus on a few number of criteria, in this paper we improve our previous approach in [4], by reducing the number of features the assessment takes into account. It is also worth noticing that, rather than assigning an article to one of the existing WikiProject classes, our goal is to evaluate the relevance of each article with respect to all the classes.

3 Assessment Results

The Wikipedia Medicine Portal community manually assesses the quality level of the published articles, to recognize excellent contributions and identify topics that instead need further work. The six quality classes are: (1) Stub, (2) Start, (3) class C, (4) class B, (5) Good, and (6) Featured article. The Featured and Good article grades are the highest assessments. They require a community consensus and an official review. All the others can be achieved with a simple review.

In [4], we assessed the maturity degree of all the articles published on the Wikipedia Medicine Portal (24,418 at the time of our study). This dataset is distinctive because it is composed of heterogeneous content, from very short drafts till comprehensive articles with a complex structure and a technical dictionary.

For the assessment, we exploited AHP [15], a multi-criteria decision making technique, which has been largely used in several fields. It helps making decisions when several different *alternatives* can be chosen to reach a *goal*. AHP is able to order the alternatives from the *most relevant* to the *less relevant*, with respect to a set of *criteria* and *subcriteria*, proceeding with a divide and conquer approach.

In our instantiation, the alternatives are the six quality classes and the output of AHP is a vector representing a new metric that we call *maturity degree*. Criteria and subcriteria of the hierarchy are quantitative features of the article and are listed below in this section. Noticeably, we do not use AHP to classify Wikipedia articles as belonging to a single class. Rather, having the maturity degree vector $v = [v_i]$, each v_i represents the relevance of the article to the corresponding WikiProject class i ($i = 1$ is Stub, 2 is Start, and so on). Similarly to the property of unimodality of a function, we say that a maturity degree vector is *consistent* when it has exactly one absolute maximum, *i.e.*, if for some value m , it is monotonically increasing for $i \leq m$ and monotonically decreasing for $i \geq m$. The property of consistency of the vector ensures that the relevance is maximal either for only one class or for neighboring classes.

Due to page limits constraints, we invite the interested reader to look at the extended version of this paper [10] for a detailed description of the AHP methodology and its instantiation leading to the maturity assessment.

In this paper, we first reduce the number of subcriteria in the AHP instantiation with respect to [4], and then we evaluate the goodness of the newly obtained maturity degree by relying on the cosine similarity (\cosSim). The \cosSim is commonly used in Information Retrieval and text mining to evaluate the similarity of two multi-dimensional vectors v_i and v_j , and it is defined as:

$$\cosSim(v_i, v_j) = \frac{v_i \cdot v_j}{\sqrt{v_i^2} \sqrt{v_j^2}}$$

Since the maturity degree of an article is always a vector with positive components, the \cosSim ranges over [0,1]. We called *cross cosine similarity* ($crcosSim$) and *class cosine similarity* ($clcosSim$), respectively, the average \cosSim between all the pairs of vectors paired with articles that on WikiProject belong to different classes, and to the same class. Intuitively, we expect that: *i*) the maturity degree vectors of articles belonging to different WikiProject classes have lower similarity than those of articles of the same class; and *ii*) the more the two classes are distant, the lower the similarity will be. Formally, given two WikiProject classes C_1 and C_2 (with $C_i \in \{\text{Stub, Start, Class C, Class B, Good article, Featured article}\}$ and $i \in \{1, 2\}$) and the maturity degree v_i of an article, then:

$$crcosSim(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{v_i \in C_1} \sum_{v_j \in C_2} \cosSim(v_i, v_j) \quad (1)$$

$$clcosSim(C_1) = \binom{|C_1|}{2}^{-1} \sum_{v_i, v_j \in C_1, i \neq j} \cosSim(v_i, v_j) \quad (2)$$

In a more formal way, if an index ranging from 1 to 6 represents each WikiProject class, we should observe that $crcosSim(C_i, C_j) < crcosSim(C_i, C_k)$ when $i < j < k$. Moreover, a $clcosSim$ higher than 0.95 would mean that our maturity degree mimics the WikiProject classification.

In the following, we report our experimental results, on the whole database of 24,418 articles. In our previous work we considered four criteria inspired by [17,22]: *lingual*, *structural*, *historical* and *reputational*. Subcriteria belonging to the *lingual* criteria were: (1) Flesch reading ease and (2) Flesch-Kincaid grade level; (3) word count and (4) sentence count; (5) multi-syllable words / words ratio; (6) spell error / words count ratio. *Structural* subcriteria were: (1) number of categories; (2) internal and (3) external links; (4) non-textual resources; (5) further readings; (6) number of symbols in title; (7) section headings count; (8) number of citations. The *historical* criterion was made by sub criteria: (1) edit counts (times that the article has been edited); (2) editor count (number of different users that edited the article); (3) number of devoted editors ratio; (4) anonymous editors ratio; (5) minor edits ratio; (6) article age; (7) edit frequency. The *reputational* subcriteria were: (1) average active age of editors; (2) average upload amount of editors; (3) average edit times of editors; (4) average talk times

Table 1. Average similarity for the articles belonging to the same WikiProject class (class \cosSim) and to distinct classes (cross \cosSim), with different settings

		average \cosSim		
considered classes		no AHP	AHP 25 subc. ([4])	AHP 9 subc.
class \cosSim	Stub - Stub	0.88	0.98	0.95
	Start - Start	0.91	0.93	0.87
	Class C - Class C	0.93	0.92	0.85
	Class B - Class B	0.90	0.92	0.87
	Good Art. - Good Art.	0.89	0.94	0.89
	Feat. Art. - Feat. Art.	0.89	0.96	0.95
cross \cosSim	Stub - Start	0.89	0.87	0.76
	Stub - Class C	0.89	0.75	0.59
	Stub - Class B	0.84	0.63	0.46
	Stub - Good Art.	0.83	0.59	0.40
	Stub - Feat. Art.	0.77	0.53	0.34
	Start - Class C	0.92	0.89	0.81
	Start - Class B	0.88	0.79	0.71
	Start - Good Art.	0.86	0.74	0.64
	Start - Feat. Art.	0.79	0.67	0.55
	Class C - Class B	0.90	0.88	0.83
	Class C - Good Art.	0.88	0.85	0.79
	Class C - Feat. Art.	0.83	0.79	0.73
	Class B - Good Art.	0.89	0.92	0.87
	Class B - Feat. Art.	0.86	0.90	0.86
	Good Art. - Feat. Art.	0.88	0.94	0.91

of editors. Hereafter, we validate *i)* the choice of using AHP for assessing the maturity of an article (Section 3.1), *ii)* the results obtained in [4] (Section 3.2), and *iii)* the refined version of the approach proposed here (Section 3.4).

3.1 Similarity of Results with 25 Subcriteria without AHP

The first experiment is aimed to verify that AHP effectively helps on making decision about the maturity assessment of a medical article. Hence, in this experiment we don't use AHP and we evaluate the \cosSim of the articles directly on the value of the 25 subcriteria. In particular, to compute the similarity of two articles, we compute the \cosSim of the two vectors reporting the values of the 25 normalized features corresponding to two articles. The average similarities are reported in the first column of Table 1. It is clear that the vectors of all the classes exhibit a very high similarity. Some classes, indeed, exhibit also higher values of cross \cosSim than class \cosSim : Start, for example, has a class \cosSim of 0.91, but a cross \cosSim of 0.92 with Class C. This experiment confirms that the straightforward approach of considering the statistic distribution of the 25 features can not produce accurate results. A better approach that takes into account a finer weight of the different features is, then, advisable in order to better deal with the statistical fluctuation of the features among the different classes.

3.2 Similarity of Results with 25 Subcriteria

The second experiment evaluates the similarity of the maturity degrees obtained in [4]. The results are shown in the second column of Table 1. Comparing the

Table 2. Mutual information of structural and historical features w.r.t. the WikiProject article class. It captures the dependency of the feature w.r.t. the class.

feature	mutual information	removed
<i>structural features</i>		
section headings count	0.59	no
internal links	0.52	no
number of citations	0.43	no
non-textual resources	0.12	yes
spell error / words count ratio	0.10	yes
external links	0.10	yes
further readings	0.02	yes
number of categories	0.01	yes
number of symbols in title	0.00	yes
<i>historical features</i>		
edit count	0.44	no
edit frequency	0.43	no
editors count	0.36	no
anonymous editors ratio	0.13	yes
article age	0.13	yes
number of devoted editors ratio	0.03	yes
minor edits ratio	0.05	yes

similarity of the normalized feature vectors with the similarity of the maturity degree obtained in [4], we can appreciably observe an increase of the class *cosSim* and a decrease of the cross *cosSim*. In particular, we notice that the class *cosSim* always has values above 0.92, meaning that the maturity degrees of the articles within the same WikiProject class are very similar among them. However, this may produce a low granularity characterization of the maturity of different articles. We also observe that, as expected differently from the previous experiment, the cross cosine similarity decreases as the two considered classes are distant among them. For example, the average *cosSim* of articles in class *Stub* decreases as the other class is more distant: $crcosSim(Stub, Class C) = 0.75$, while $crcosSim(Stub, Feat. Art.) = 0.53$.

3.3 Reducing the Number of Subcriteria

Many studies have shown that AHP performs better with few criteria [11,15]. Here, we reduce the number of criteria having a twofold beneficial effect: it speeds up the decision making process and improves its results in terms of quality.

To reduce subcriteria, we consider several aspects: adhesion to the Wikipedia guidelines and reduction of redundancy. We eliminate the whole reputational criteria, not considered by the guidelines. Then, we adopted the *mutual information* measure (or *information gain*) to evaluate whether a feature brings more information. This measure evaluates the dependency of two random variables: the mutual information $I(X; Y)$ represents the reduction in the uncertainty of X due to the knowledge of Y [5]. It is defined as $I(X; Y) = H(X) - H(X|Y)$ where $H(X) = -\sum_x p(x) \log p(x)$ is the usual definition of the *entropy* of a random variable X and $H(X|Y)$ is the *conditional entropy* of X given Y [8].

For the lingual criterion, we started removing spell error, because of its bias against complex and composite words, typical of the medical terminology. Then,

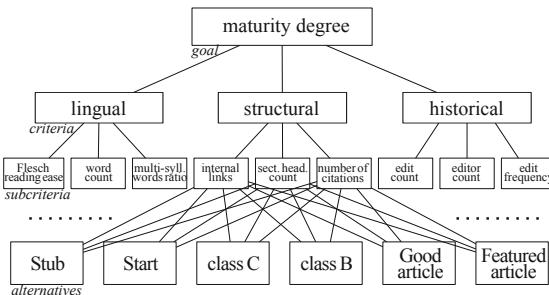


Fig. 1. AHP hierarchy with 9 subcriteria

we noticed that some of the lingual subcriteria actually consider the same property of an article. In particular, word count and sentence count are tightly related one with each other, considering the length of the article. Similarly, Flesch reading ease and Flesch-Kincaid grade level both consider the complexity of the articles. Considering both sentence count and word count has the effect of doubling the influence of the article length property on the final outcome. The same happens for the two features that consider the article complexity. Then, we removed the sentence count and the Flesch-Kincaid grade level, since their mutual information with word count and Flesch reading ease (namely the amount of information gained about Y after observing X) is very high, 0.83 and 1.39, respectively, and this would lead to overestimate the same property.

To decide which of the subcriteria belonging to the structural and historical criteria should be removed, for each article we computed the mutual information of each criterion w.r.t. the article class. In this way, we identified a set of subcriteria that do not provide any help to the decision process, namely those that exhibit a uniform distribution of the articles among all the WikiProject classes, with the lowest mutual information w.r.t. the article class. In particular, the removed features are reported in Table 2, jointly with their mutual information w.r.t. the article class: the more the value is close to 0, the more the feature is unrelated with the class assigned by WikiProject. Such subcriteria, actually, only introduce random noise and, consequently, complicate the decision process.

The procedure leads us to only 9 subcriteria: 1) Flesch reading ease (FR), 2) word count (WC), 3) multi-syllable words / words ratio (MS), 4) internal links (IL), 5) section headings count (SHC), 6) number of citations (NoC), 7) edit count (EditC), 8) editors count (EditorsC), and 9) edit frequency (EF).

3.4 Maturity Assessment with 9 Subcriteria

We apply AHP with 9 subcriteria and re-compute the maturity degree of the articles of our dataset (see Figure 1). We remind the reader that the full AHP instantiation leading to the maturity degree is described in [10]. Technically, the steps leading to the results are available online at <http://goo.gl/1i6CpI>.

Table 3 compares the consistency (as defined at the beginning of this section) of the new results with the old ones. Firstly, we observe that a slightly higher consistency ratio holds for the new results (it is almost perfect for Stub articles).

Table 3. Percentage of articles that obtained a consistent maturity degree

WikiProject Class	consistent maturity degrees	
	25 subcriteria	9 subcriteria
Stub	95%	99%
Start	98%	98%
Class C	94%	93%
Class B	88%	87%
Good Article	94%	96%
Featured Article	86%	88%

The third column of Table 1 reports the class and the cross similarity for the new results. It is evident the reduction of cross similarity between the different classes, but also a reduction of the class similarity. We can consider those two phenomena as two different beneficial effects. Firstly, the cross similarity reduction happens because the new assessment it is able to better discriminate between articles belonging to different WikiProject classes, better capturing the varied maturity degrees within the classes. Secondly, the sensible increase of the class \cosSim means the ability to more precisely characterize the articles belonging to the same class, providing finer and more specific levels of maturity. This can be ascribed to the reduction of redundancy and to the adoption of the intervals, as described in details in [10]. With a high redundancy, features that capture the same aspects (like the article length in case of words and sentences count) overemphasize them, vanishing the smaller differences introduced by the features that consider other aspects. When the overemphasized aspects are also considered more relevant during the AHP process, this effect of flattening is further stressed. Another inherent contributing factor is the adoption of the intervals: articles with two subcriterion values falling within the same interval but near to its two opposite ends, have more similar final maturity degrees. This flattening effect is reduced with the new assessment, since the results for the articles within the same class are slightly different among them, leading to a finer granularity of the maturity degrees.

Figure 2 shows a summary of some results of the new assessment, in order to give a glance of the obtained maturity degrees. The figure does not intend to detail the results for each of the analyzed articles, but only to highlight the general trend of the results for 300 considered articles within some of the different classes. In particular, for each WikiProject class, we randomly sample 50 articles belonging to that class and draw their resulting maturity degree as a line following the relevance of each class. As for the original work, we have some articles with a maturity degree significantly different from the WikiProject class they belong to. For example, in Figure 2(f) that shows the results for Featured Articles, we can notice a couple of articles that have their maximum relevance on the C quality class: this reflects the fact that the manual assessment by WikiProject considers also qualitative guidelines, as the neutrality and the comprehensiveness, hard to compute in a quantitative way.

Summarizing, reducing the subcriteria set leads us to an efficient application of AHP, as discussed in [11,15]. Further, the new set of criteria yields a finer

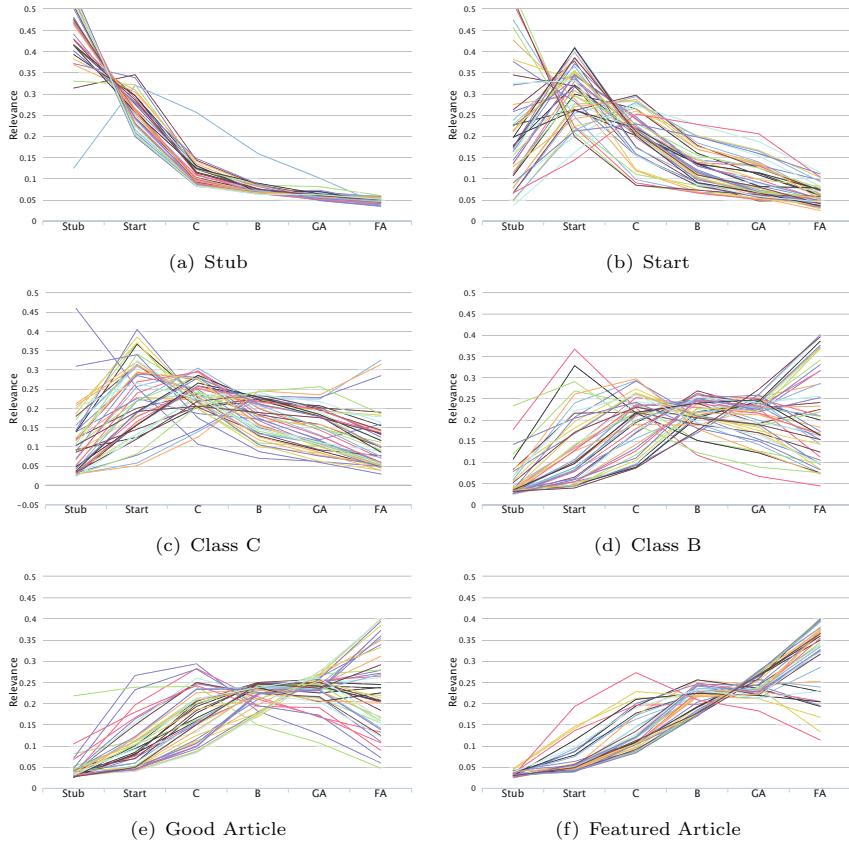


Fig. 2. Maturity degree with respect to the WikiProject assessment, 9 subcriteria. Each line is the maturity degree of an article belonging to a given WikiProject class.

assessment of the articles belonging to the same WikiProject classes and a more evident separation between the articles belonging to different classes.

4 Conclusions

This paper enhances our previous automatic assessment of Wikipedia medical articles by identifying and pruning redundant features. In this way, we obtained fine-grained results to evaluate the relevance of each article w.r.t. the WikiProject classes. To validate our results, we computed the similarity of each pair of articles exploiting cosine similarity. We observed that, with a reduced set of features the average cross-similarity of articles (those belonging to two distinct classes) is lower, leading to a more evident separation into classes. The average class-similarity for articles of the same classes is also lower, yielding a finer intra-class assessment. We conclude that the new assessment with the reduced

set of features better discriminates the articles, since their evaluation is, at the same time, closer to the one given by WikiProject and fine-grained, with the added-value of an automatic process behind the evaluation outcome.

References

1. Aitken, M., Altmann, T., Rosen, D.: Engaging patients through social media. Technical report, IMS Institute for healthcare informatics (January 2014), <http://goo.gl/BoFJA8> (last checked: August 27, 2014)
2. Blumenstock, J.E.: Automatically assessing the quality of Wikipedia. School of Information Report 2008-021, UC Berkeley (2008)
3. Blumenstock, J.E.: Size matters: Word Count as a measure of quality on Wikipedia. In: 17th International Conference on World Wide Web (2008)
4. Conti, R., Marzini, E., Spognardi, A., Matteucci, I., Mori, P., Petrocchi, M.: Maturity assessment of Wikipedia medical articles. In: CBMS 2014. IEEE (2014)
5. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience (2006)
6. Dalip, H., et al.: Automatic quality assessment of content created collaboratively by web communities: A case study of Wikipedia. In: 9th ACM/IEEE-CS Joint Conference on Digital Libraries. ACM (2009)
7. Haigh, C.A.: Wikipedia as an evidence source for nursing and healthcare students. Nurse Education Today 31(2) (2011)
8. Hamming, R.W.: Coding and Information Theory, 2nd edn. Prentice-Hall, Inc., Upper Saddle River (1986)
9. Hu, M., Lim, E.-P., Sun, A., Lauw, H.W., Vuong, B.-Q.: Measuring Article Quality in Wikipedia: Models and Evaluation. In: 16th CIKM. ACM (2007)
10. Marzini, E., Spognardi, A., Matteucci, I., Mori, P., Petrocchi, M., Conti, R.: Improved automatic assessment of Wikipedia medical articles. Technical report, IIT-CNR, TR-IIT-11-2014 (August 2014)
11. Miller, G.A.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review 63 (1956)
12. Mobihealthnews.com. Almost 1M families use video consultations with physicians last year (March 2014), <http://goo.gl/6sI0tD> (last checked: August 27, 2014)
13. Feltman, R.: The #1 doctor in the world is Dr. Wikipedia. MSN News online (January 2014), <http://goo.gl/zhTN0a> (last checked: August 27, 2014)
14. Fox, S., Duggan, M.: Health Online 2013. Pewinternet.org (January 2013), <http://goo.gl/vRBnCA> (last checked: August 27, 2014)
15. Saaty, T.L.: Decision making with the Analytic Hierarchy Process. Int. Journ. of Services Sciences 1(1) (2008)
16. Sillence, E., Briggs, P., Harris, P.R., Fishwick, L.: How do patients evaluate and make use of online health information? Social Science and Medicine 64(9) (2007)
17. Stvilia, B., Twidale, M.B., Gasser, L., Smith, L.C.: Information quality discussions in Wikipedia. In: Knowledge Management (2005)
18. Suzuki, Y., Yoshikawa, M.: Assessing Quality Score of Wikipedia Article Using Mutual Evaluation of Editors and Texts. In: 22nd CIKM. ACM (2013)
19. U.S. Dept. of Health and Human services. Online Health Information: Can You Trust It?, <http://goo.gl/G5evGg> (last checked: August 27, 2014)

20. Warncke-Wang, M., Cosley, D., Riedl, J.: Tell Me More: An actionable quality model for Wikipedia. In: 9th Intl. Symposium on Open Collaboration. ACM (2013)
21. Wilkinson, D.M., Huberman, B.A.: Cooperation and Quality in Wikipedia. In: Intl. Symposium on Wikis. ACM (2007)
22. Wu, K., Zhu, Q., Zhao, Y., Zheng, H.: Mining the factors affecting the quality of Wikipedia articles. In: Inform. Science and Management Eng., vol. 1 (2010)
23. Xu, Y., Luo, T.: Measuring article quality in Wikipedia: Lexical clue model. In: 2011 3rd Symposium on Web Society, SWS (2011)

Integrity Management in a Trusted Utilitarian Data Exchange Platform

Sweety Agrawal, Chinmay Jog, and Srinath Srinivasa

International Institute of Information Technology Bangalore,

26/C, Electronics City, Bangalore India

{sweety.v.agrawal,jog.chinmay}@iiitb.org,sri@iiitb.ac.in

<http://www.iiitb.ac.in>

Abstract. Utilitarian data refers to data elements that can be readily put to use by one or more stakeholders. Utility of a data element is often subjective and intertwined in the sense that, a positive utility for one stakeholder may result in a negative utility for some other stakeholder. Also, credibility of utilitarian data is often established based on the credibility of its source. For this reason, defining and managing the *integrity* of utilitarian data exchanges is a non-trivial problem. This paper describes the problem of integrity management in an inter-organizational utilitarian data exchange platform, and introduces a *credentials*-based subsystem for managing integrity. Scalability is addressed based on mechanisms of privilege percolation through containment. Formal characteristics of the proposed model are derived based on an approach of adversarial scenario-handling.

Keywords: Utilitarian Data, Integrity Management, Credential-based Privileges, Access Control.

1 Introduction

Utilitarian data refers to data elements that can be readily put to use by one or more stakeholders. An example of this would be data about prices of agricultural produce in different markets of a district. Several community-oriented applications have been explored to share utilitarian data among its members. However, several shortcomings persist due to lack of clarity in our understanding of data utility.

Utilitarian data has several unique properties. Firstly, the validity of a utilitarian data element is “bounded” within specific contexts. For example, a sentence like *PricePerKilogram(Sugar, 50)* that can be expressed by an RDF triple, may not be valid in a context-free fashion. Its validity may be limited to the context of a particular marketplace or a vendor.

Data utility is not an objective property of the data itself, but a subjective relationship between data elements and stakeholders. What is utilitarian to one stakeholder may not be utilitarian to another. These relationships may also have dependencies across them. A positive utility to one stakeholder, may result in

a negative utility to some other stakeholder. For example, accessing a person's medical history from a hospital, may have a positive utility for an insurance company, but a potential negative utility (in the form of increased premium) for the person who is applying for insurance.

The utilitarian value of data, also makes the issue of credibility of the data, important. Not all utilitarian data is publicly known and can be aggregated using open strategies like crowdsourcing. This makes the credibility of utilitarian data, dependent on the credibility of its source. Source credibility or "trust" also plays a role in trusted exchanges of utilitarian data. For example, a stock broker may not accept a stock tip from a random person, but may be willing to put all the bets if the tip comes from a trusted contact.

Based on this, we classify utilitarian data exchanges into three categories – *open* (trusted and untrusted), *open-ended* and *closed* utilitarian data.

Open utilitarian data is publicly available. The owner of this data has no knowledge or control over who can use this data. An example of this is Open Data initiatives by several governments worldwide. This kind of a system can be represented as in Figure 2. Such data is also "trusted" because the consumers of these data are aware of and trust the governmental processes that acquire and publish such data. In some cases like WikiTravel¹, utilitarian data may be shared by anyone, making it difficult to establish the veracity of the data. We term it as "untrusted" open data. This kind of a system can be represented as in Figure 1.



Fig. 1. Untrusted Open Data

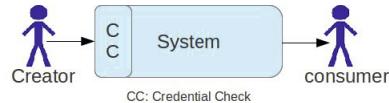


Fig. 2. Trusted Open Data

Closed utilitarian data refers to utilitarian data that is private. The owner of this data has exact knowledge and control over how this data is exchanged. Password for an organizational email account, or PIN for a shared credit card are examples of closed utilitarian data. This kind of a system can be represented as in Figure 4.

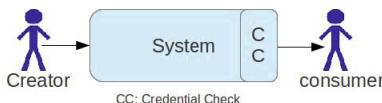


Fig. 3. Open-ended Data

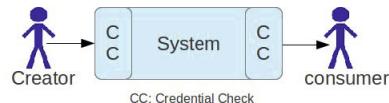


Fig. 4. Closed Data

But there is a class of utilitarian data which are neither publicly available, nor strictly private. Here, the data owner shares data to be disseminated within a trusted system. Once the data is shared, the owner has no control or knowledge of who accesses it – just as in open data. However, the system provides a guarantee

¹ http://wikitravel.org/en/Main_Page

that the data will only be accessed by legitimate users. The “system” here need not be a single organization, and may involve several organizations. For example, the national secondary school education board in several countries provide a service for colleges to verify grades, credentials and other details of registered students applying for admission. In this case, students have shared their details with the education board, but they may not know about which colleges are accessing their records. But they trust the education board for sharing this data legitimately. We term this kind of a system as an “open-ended” data exchange platform, and is represented as in Figure 3.

This work focuses on open-ended utilitarian data exchanges across multiple organizational settings. To represent utilitarian data, the authors had earlier proposed a framework based on Kripke Semantics, called Many Worlds on a Frame (MWF) [16]. We extend this work to address the problem of integrity in data access and exchange, when the MWF model is implemented across several organizations. The outcome is a credentials-based subsystem and a logic to reason about data integrity based on credentials.

2 Related Literature

Several models of access control exist in the literature. In the simplest form, access privileges are associated directly with users, on establishing their identity. We call this as *identity-based* access control.

Such a method is not scalable over large inter-organizational data sharing platforms. Given the number of users and the churn in user affiliations, associating privileges with users directly is both tedious and risky.

The term “credentials” is often used in identity-based access control as a means of establishing user identity, based on certification by trusted third-parties [1,10,17]. However, the privileges are still associated with the identity of users. In contrast, the focus on this work is to safely establish privileges with credentials themselves. Any user satisfying those credentials should be able to obtain the said privileges.

Another approach for managing privileges over an open infrastructure is based on *self-descriptions* by data elements [2]. Here, the aim is to address privacy concerns by having individual entities describe their own access control mechanisms. While this work has several similarities and is relevant to our problem, it does not address the semantics of access control based on how entities are related to one another.

Yet another form of access control is called *relationship-based* access control [7]. In this mechanism, authorization is based on the relationship between the owner and consumers of a resource. It also captures the context in which these owner-consumer relationships are defined. Relationships are also seen as a means of delegation of trust.

While this work addresses an important requirement for our problem, there is no notion of credentials per se.

One of the models receiving the most widespread research attention for scalable access control mechanisms is the *Role Based Access Control* (RBAC)

model [3,8,11,14]. Here, access privileges are assigned to *roles*, rather than users. Any user playing a given role obtains the privileges assigned to that role.

There is a subtle but important difference between RBAC and the credentials-based model proposed in this paper. Definitions of roles and assignments of roles to users is defined within the framework of a larger organizational context. Specifically, assignment of roles as well as assignment of access privileges are in the purview of the same organizational entity. However, in our work, we have to address data exchanges spanning several organizations. Each organization would have its own systems of roles and role assignment policies. In such a system, the *credentials* of a user is defined as the roles played by the user across *all* organizational contexts. Access privileges are based on assertions on credentials rather than roles. This means that access privileges are based on elements that are outside the control of the organizational entity creating the access privileges.

For instance, an organization called *Lakshmi Nursing Home* of type *Hospital* may share its casualty related medical records by allowing viewing of the data by any user u who satisfies the following credentials: " u plays the role of a *Doctor* in any organization of type *Hospital* located in *Karnataka*, which is a *State* contained in *India*." Here, access privileges in a given organization is established based on role assignments made in some other organization.

When multiple organizations can independently assign roles, managing the veracity of credentials becomes the primary research challenge in the proposed model.

Another access control mechanism called *capabilities* is proposed in [9]. The author of this work points out the limitations of RBAC and how it would not possible to create roles and assign permissions to these roles to access web-pages. They propose capabilities where a user can possess capabilities only after it is given to her. But, the problem we are trying to address here is different in the sense that we know the credentials of a user and wish to restrict the access control based on the user's credentials. Credentials are not assigned by the proposed system, instead the credentials of a user are formed by her participation in various organizations. We only try to model the credentials and the organizations can use these credentials to control access.

Yet another paradigm of access controls is *profile-based* access controls, that determine access privileges based on activity patterns of users [4]. Here access is provided to the user on the basis of thresholds established by the System Security Officer depending upon the access pattern of the user. If the user seems to be shifting from her usual access pattern, the threshold will be reduced to a lower trust level as a precaution. Access requirements based on context requires the access control models need to be aware of the context. X-GTRBAC [5], an XML based access control is extended in this work. The roles are assigned in this model by the "trust" factor. "Trust" is linked with a user as a confidence on the basis of the certified attributes. The confidence is obtained from the certification provided by the Trust Management approach of third parties.

The authors of [18] extend profile-based access control to obtain activity information from social network data. The tool uses mining techniques over social network structures to define social roles and their privacy settings.

While profile-based access controls are more flexible and scalable than identity-based access controls, our requirements need specific entitlements in the form of credentials in order to define access privileges.

Risk aware RBAC [6] is yet another model for access control. In this work, when a request is made by a user to access a resource, the request is analyzed to estimate the risk of granting or denying access to a user. Even a risky access is granted if the risk is not high. For this reason, risk estimation and management is an important step in this work. The risk analysis is done at the permissions level.

We do not have a notion of quantification of access risk in the proposed model. However, there is an inherent risk associated with credentials-based access control. Since credentials are provided by third parties, the credibility of the credentials are always in question. We address this question by asking how easy it would be for an adversary to create fake credentials to gain access.

3 Many Worlds on a Frame

The proposed model is built over a knowledge representation framework, Many Worlds on a Frame (MWF) [15,16]. MWF model was proposed as a means for representing utilitarian knowledge. This framework is loosely based on Kripke semantics [12]. The MWF model comprises of several conceptual “worlds” within which data is described. The worlds themselves can also be used as data elements in other worlds. The MWF is well suited for implementing an inter-organizational data exchange platform, where each participating organization would itself feature as a conceptual world. In this section, we introduce the essential elements of the MWF framework.

Conceptual Worlds and the Frame MWF describes data elements and their relationships with other data elements, in a bounded context called a *Conceptual World* or simply *World*. A world is not just a name space, but also functions as an active data element.

Every conceptual world has a *type* and a *location*, which are other worlds themselves. The type construct represents an “is-a” relationship between worlds. The location construct represents an “is-in” relationship. The interconnection of worlds using the type and location constructs forms the global data structure called the *Frame*. Formally, a frame is defined as

$$F = (C, T, L) \quad (1)$$

Here, C is a set of concepts or worlds. $T : C \rightarrow C$ represents the type of a world and $L : C \rightarrow C$ represents the location of a given world.

The semantics of type and location are reflexive, antisymmetric and transitive, with a greatest element. Thus, each hierarchy is a rooted acyclic graph. The root

of the type hierarchy is a world called *Concept*. The root of the location hierarchy is a world called *Universe of Discourse(UoD)*. This means that every other world is sub-classed from *Concept*, and is located in the *UoD* where,

$$T(\text{Concept}) = \text{Concept} \quad (2a)$$

$$L(\text{Concept}) = \text{UoD} \quad (2b)$$

$$T(\text{UoD}) = \text{Concept} \quad (2c)$$

$$L(\text{UoD}) = \text{UoD} \quad (2d)$$

For any world, its “is-a” and “is-in” ancestry is called its *lineage*.

A conceptual world can either be a *class* or an *instance*. A class can be sub-typed using the type hierarchy. An instance cannot be sub-typed, but it can host data elements. Both classes and instances can contain other classes and instances.

Semantics of the type and location hierarchies The type hierarchy inherits the structure of a world and its attributes, also called its schema. A world lower down in an is-a hierarchy can override a structural element like Role or Association by defining another structural element by the same name.

The location hierarchy is used to reason about privileges and visibility. The semantics of a location hierarchy are not that of inheritance per se, but of *availability*, therefore, all the worlds that are located inside a world c are *available* to the world c . It also means that all the users who have privileges on c get at least as much privileges on all worlds located inside c . This way, the privileges percolate through the location hierarchy.

Structure of a World Every class has a structure or a *schema*, and an instance of a class also hosts *data* according to its schema.

In order to obtain a vocabulary for describing data and structural elements, a world “imports” other classes from the Frame. Classes and subclasses of imported worlds can be used to describe the schema of the importing world. Instances of imported worlds can feature as data elements inside the importing world. The existence of an imported class or an instance is “visible” inside the importing world. However, they may not necessarily be “accessible” by users of the world as the access to a world depends on satisfying credentials.

There are two elements to the schema hosted by a world: *attributes* and *relationship types*. Analogously, *attribute values* and *relationship instances* contribute to the data hosted by a world instance.

Attributes of a concept are represented by (*name, type*) pairs. The type of an attribute can be one of several basic types described in MWF such as, string, URL, date and number or *type* can refer to one of the imported classes in the world.

Relationship types are of two kinds - Role and Association. The first type, a *Role* or a *Component* represents a semantic building block of the world. A role is said to be “played” by a *role player*, which can be any of the imported classes.

For instance, suppose we have a conceptual world called *School*, that has imported another class called *Person* from the Frame. A role called *Teacher* can now be defined inside *School* with *Person* as the role-player. Similarly, a class called *AcademicActivity* can be created within *School*, to play the role of a *Course* inside *School*.

The second form of relationship type, called an *Association*, is a relationship defined between two roles. An association is represented as a $(\text{subject}, \text{predicate}, \text{object})$ triple, where *subject* and *object* are roles defined in the world and *predicate* is the association name. For instance an association $(\text{Teacher}, \text{TeacherOf}, \text{Course})$ can be designed to connect the two roles defined above.

Both roles and associations are allowed to have any number of attributes represented by $(\text{name}, \text{type})$ pairs, that augment their descriptions.

Using these imports, schema and data, we can capture the essential features of a utilitarian world. This is schematically shown in Figure 5. *School*, an example

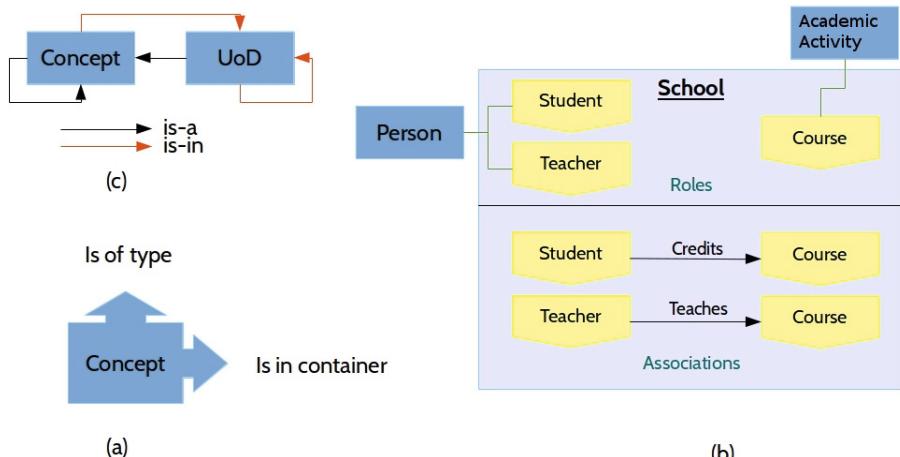


Fig. 5. Many Worlds on a Frame

of a world class, is depicted in Figure 5(b). In the example, the concept of *Person* plays two *roles* - a *Student* and a *Teacher*. Similarly, *AcademicActivity* plays the role of a *Course*.

Users and Consumers Users of an MWF knowledge base are modeled as an integral element of the dataset itself. This helps in representing dependencies and constraints across classes of knowledge creators and consumers. Access privileges for users are determined on the basis of their memberships in other utilitarian

worlds. Memberships are nothing but the participations of the users as role-players in these worlds.

Every MWF installation contains a world called *Person* by default. Every user of the system is represented in a world of type *User*, which is of type *Person*. Users can thus be featured as role players in worlds that import them. The “participation set” or “credentials” of a user u , denoted as $\text{participations}(u)$, is a set of tuples of the form $(\text{role}, \text{world})$ detailing all the worlds and the corresponding roles played by u anywhere in the Frame.

Consumers represent two classes of people: the set of all users who feature as worlds in the system, as well as any visitors to the system. A casual visitor to the system will have null credentials, which will be matched against privilege rules like any other user.

Privileges Different levels of access to a world are defined by a system of *privileges*. There are five levels of privileges: *Privilege level*, *Frame level*, *Schema level*, *Data level* and *Visibility level*. The function of each of these levels is described in our previous study [16]. Privilege level provides privileges to add, modify or delete access rules in a world. Frame level provides privileges to modify the lineage of a world, that is, changing “is-a” and “is-in” parent of a world. Schema level gives privileges to add, remove or modify the roles and associations inside a world. Data level privileges are to add the data instances for roles and associations and values of attributes. Visibility level privileges govern whether a world as a whole is visible to a user or not.

Depending on the credentials of a consumer, a privilege “level” can be assigned to her. This is specified by rules written as Horn clauses [13] of the form:

$$\text{PrivilegeLevel} \leftarrow \text{Assertion}_1, \text{Assertion}_2, \text{Assertion}_3, \dots, \text{Assertion}_n \quad (3)$$

Assertions are of the form $(\text{Role}, \text{Type})$. The assertion can be read as: credentials of consumer should have a role *Role*, in a world of type *Type*. In a given rule of the form shown above, *all* assertions have to be satisfied by the consumer to gain the particular privilege level.

The rule discussed above is proposed in [16]. In the proposed work, we have modified the access rules to accommodate the location of a world as it is important to grant privileges. In the next section, we elaborate the integrity management subsystem of MWF that builds on the credentials and privilege mechanisms.

4 Integrity Management

MWF is designed to be an open platform for exchanging utilitarian data. A given MWF Frame would span across several participating organizations. Given this, it would be infeasible to associate access privileges to users directly. The integrity of data access and exchanges are established based on the *credentials* of the user performing the operation.

Credentials of a user, as defined earlier, is the set of all the participations of a user in different worlds. The credentials for a user u can be defined as

$$C_u = \{(r_1, w_1), (r_2, w_2), \dots, (r_n, w_n)\} \quad (4)$$

where, r is a role defined in world w and played by user u .

In MWF, a world can have five modes of access privileges as defined in 3: Visibility, Privilege management, Frame management, Structure management and Data management. A *privilege package* defines the privileges that are available to a user on a particular world. Therefore, a privilege package can be defined as a 5-bit binary string, where each bit corresponds to a privilege mode.

An access rule can be defined in a world w in the form shown below:

$$\text{PrivilegePackage}_i \leftarrow r_1(t_1, l_1) \wedge r_2(t_2, l_2) \dots \wedge r_n(t_n, l_n) \quad (5)$$

Table 1. Privilege Package definition

Bit no	Level	Value	Definition
1	Visibility	1	can view the structure, data, privilege rules, lineage of a world
		0	can not view a world (disables rest of the privilege modes)
2	Privilege	1	can add/modify access rules (excluding inherited rules)
		0	not allowed to add/modify access rules
3	Frame	1	can modify type and location of a world (thus affecting world structure and access rules)
		0	not allowed to modify type and location of a world
4	Structure	1	can add/modify roles, associations, attributes of a world
		0	not allowed to add/modify roles, associations, attributes of a world
5	Data	1	can add/remove role and association instances from a world instance
		0	not allowed to add/remove role and association instances from a world instance

An assertion of the form $r(t, l)$ says that the user trying to access this world should be playing a role r in a world of type t located in or contained in l . On satisfying all the assertions on the right hand side of such a rule, the user obtains *PrivilegePackage* _{i} which is a 5-bit string.

For example, the following rule:

$$10001 \leftarrow \text{Principal}(\text{College}, \text{India})$$

provides visibility and data access privileges for any user who is playing the role of *Principal* in any world of type *College*, located in or in a world contained in the world *India*.

Each bit in a privilege package represents a privilege mode which is either permitted or prohibited. The definition of every bit in a 5-bit string is shown in

Table 1. Thus, if a user gets a privilege package of 11000 on a world w , then the user can view the world and has privileges to add/delete access rules in a world w .

If the visibility privilege mode for a world is unset, then all other bits in the privilege package are ignored.

Also, as noted earlier, access rules percolate through the is-in hierarchy in the Frame. If a world c is contained in world w , all access rules of w will also apply for c . A user having a privilege package p_i on w can also use the same privilege package in c .

When the lineage of a world is changed, it not only affects the world's structure based on its new location in the is-a hierarchy, but also affects the set of privilege packages that a user can use, based on its new location.

The *privilege mode*, or the second bit in the privilege package of a world, allows changes only to access rules defined in that world. Access rules, that have percolated from container worlds above, cannot be altered by a user even if the user has privilege mode access on this world.

In a world, several access rules may be defined each granting different privileges. A user visiting a world may satisfy more than one access rules. In such cases, the privilege package assigned by all the satisfied rules will be available for the user. The user needs to *activate* any one of the available set of privilege packages. Logging of user activities for purposes of provenance, will also record the active privilege package under which the logged action was performed.

Containment and Visibility: Consider a world c contained in a world w . Since all privilege packages available to a user in w will be available on c as well, the user's privileges on c will be no lesser than that on w .

However, when it comes to visibility the above condition gets more stringent. If user u does not satisfy any access rule on w that provides visibility on w , then the user will not have visibility on c even if there is some access rule in c that provides visibility to u . If the contents of a world are invisible to a user, then all contained worlds inside it are also invisible.

Hence access rules for a world w is computed for user u only if there exists at least one rule in each level of the is-in ancestry where u will obtain visibility mode privileges.

Default rule: A default access rule is a rule of the form:

$$\text{PrivilegePackage}_d \leftarrow \perp \quad (6)$$

Here \perp refers to the “null” assertion which is satisfied by any consumer, including visitors who are not users of the system. A default rule specifies the minimum set of privileges available for a world. Typically this will be of the form (10000) indicating only visibility.

It is not necessary to define a default rule for every world. Since access rules percolate through the is-in hierarchy, a default rule defined for a world will be available to all worlds contained in it. A default rule defined for the *UoD* which

is the root of the containment hierarchy, will define the default access privileges for all worlds in the knowledge base.

A world that provides visibility by default is said to be an *open* world, while a world that forbids visibility by default is said to be a *closed* world. We can see that any world that has at least one non-default access rule is an *open-ended* world.

Overriding privileges: Unlike inheritance of structure with the is-a hierarchy, privileges are said to “percolate” and are not “inherited” through the is-in hierarchy. This means that privileges cannot be overridden by lower level classes. Suppose that in a world, an assertion set A provides privilege package P_A , and in a contained world, assertion set A provides another privilege package Q_A . Then for a user visiting the contained world satisfying assertion set A , will have both packages P_A and Q_A enabled. The user may *activate* any of the packages to perform a given operation.

The reason why access rules are not overridden is to enable clarity in provenance logging. Combining privilege packages may be risky as it may lead to creation of a privilege that was not intended.

There is an exception to privilege overriding, when it comes to default privileges. A default privilege can be overridden in a contained world by redefining the privilege package for the \perp assertion.

Hence, a closed world can be contained inside an open world. Similarly, a closed world may contain an open world that will provide default privileges to all users who have visibility privileges on the container.

Administrators: While much of the access privileges are mapped to user credentials, there are a small set of maximal privileges that are still mapped to users by identity. These are privilege packages of the form (11111) assigned to specific users called *world administrators* with an exception of viewing/modifying data elements. This kind of identity-based mapping is necessary for the following reasons:

1. Initialization of a new MWF Frame requires creation of basic worlds like *Concept*, *UoD*, *Person*, etc. Since no credentials are yet defined, access to these worlds have to be based on user identity
2. Bugs in access rule specifications or changes in role structures in some other part of the Frame may lock out a world from all users. Such situations require administrator access to repair access for the world.

Every world will maintain an “admin list” which details the set of administrators for the world. The creator of the world will be automatically added to the admin list. Admins in the admin list may add or delete users to and from the admin list. Other users not a part of admin list may also obtain administrator-like privileges from a privilege package. However, such users are still not a part of the admin list and cannot modify the admin list.

Inherited administrators and notifications: Just like privilege rules, admins of a container world also has admin privileges on all contained worlds by virtue of the containment semantics. Each world will have two kinds of administrators:

admins from the “is-in” parent and local admins. Notifications about the events inside a world like: change of lineage, add/modify relationships, add/remove attributes, etc., usually goes to the local admin. If notifications to local administrators fail, the notifier moves up the is-in hierarchy to notify admins at a higher level.

Privileges of inherited admins can not be overridden by a local administrator. The only way privileges of inherited admin can be revoked is by changing “is-in” parent.

Rule Quarantine: Consider an access rule of the form

$$P_i \leftarrow r(t, l)$$

which assigns privilege package P_i to any user playing the role r in a world of type t located in or contained in l . The above rule may become “unverifiable” due to various reasons. The role r may be removed from t ; the role r may be inherited into t and t changes its is-a lineage; or l or one of its containers becomes a closed world.

In such cases, the access rule is “quarantined” and a notification will be triggered to the administrators of the world. A rule quarantine process is triggered when a user tries to access a world and it is found that at least one of the assertions of a rule can never be verified.

Deleting the access rule rather than putting them in quarantine won’t be correct semantically as the rule at some point of time did grant privileges but now it doesn’t, either because of some modification of the role or maybe the credentials to get this privilege itself might have been modified. Also, keeping information of all the access rules that gave privileges in the past helps to manage provenance.

5 Adversarial Scenario Handling

In order to measure the efficacy of the proposed credentials-based model for access control, we propose a method of “adversarial” theorem proving. Here, various adversarial attacks are envisaged that aim to undermine the integrity of access control and we evaluate how easy would it be for an adversary to implement these plans.

An integrity violation occurs when the system allows any of the following to happen:

1. Allow access to an illegitimate user with fake credentials
2. Forbid access to a legitimate user with genuine credentials

We explore several scenarios by which an adversary could create fake credentials and evaluate how easy it would be to create such scenarios. By an “adversary” we do not necessarily mean a malicious user. It simply represents the worst-case abstraction of credentials and users that could threaten access integrity.

Scenario 1. *Alice has privilege-mode privileges on world w. Alice also has frame-mode privileges on world w₁ which is contained inside a world x. By virtue of her frame-mode privileges, Alice now changes the “is-in” parent of world w₁ from x to w. Alice now gets privilege-mode privileges on w₁ which percolates from w. Alice may have lost frame-mode privileges on w₁ by this move, but it can be easily restored by adding an access rule giving herself frame-mode privileges.*

Scenario Handling: While the above set of operations may be unusual or anomalous, it cannot be concluded to be a malicious action on the part of Alice. Changes in the lineage of a world results in notifications sent to appropriate admins. In this case, either of the following conditions will hold: (a). the administrator of w is notified of this action and can decide suitably, or (b). the administrator of w is Alice herself, in which case, this is a case of Alice moving a conceptual world into a local workspace, which in itself is not a malicious action. □

Scenario 2. *Bob has frame-mode privileges on a world x but not privilege-mode privileges. Now, if Bob wants to (say) remove visibility to casual visitors for world x, it will not be possible. But, Bob can create a new closed world w and move x to be contained in w by using his frame-mode privileges on x, thus making x invisible too for casual visitors. The user u can change the “is-a” parent of a world x which may affect the access rules.*

Scenario Handling: In this case again, malicious intent cannot be established just by the sequence of actions. If Bob creates a new closed world w, it can be done only inside another world that allows Bob to have frame-mode privileges. All admins of these container worlds are notified when Bob creates w, which can be the cue to act if in case this is indeed a malicious operation. □

Scenario 3. *World w provides data-mode privileges to users who are playing a role of Teacher in any world of type College. Alice does not have data-mode privileges on w. She then creates a world AliceAcademy whose is-a parent is College and places it in a world owned by herself. She then places herself as a Teacher in AliceAcademy to get data-mode privileges in w.*

Scenario Handling: This scenario depicts a case of looseness in access rule specification. The access rule here did not specify any location constraints for the role player, which was exploited by Alice by creating a world of type *College* inside a world that she controls. While such an attack cannot be prevented, the following measures help in catching up: provenance logging on w that logs the actual roles matched by an access rule; and notification to the administrators of *College* when a new subclass is created. □

Scenario 4. *World w requires users to be playing a role of Doctor in a world of type Hospital located in world India. Bob has frame-mode privileges on a world BobMedics which is of type Hospital located in a world contained in India. Users who play the role of Doctor in BobMedics routinely visit w. Bob now changes the is-in parent of BobMedics to some other world that is not contained*

within India. Users who were Doctors in BobMedics will no longer have access to w, unless they satisfy the role from some other world.

Scenario Handling: The above scenario represents a desirable feature of the proposed system, rather than a bug resulting in a false negative. Moving the location of *BobMedics* usually has semantic repercussions (like applicable law and jurisdiction), which is what is captured by the access rule. Users who are no longer able to access *w* will get an error message following which, they can resolve the issue with Bob. \square

Scenario 5. *Alice has admin privileges on a world w but is not an admin list member of w. Alice is an admin list member of world x which is contained in her own world. Alice can change the “is-in” parent of world w to x. By this change, she becomes an admin of w and she can remove all the admins from the world and remove all the access rules.*

Scenario Handling: The above scenario represents the case where Alice tries to lock out a world from any access. All the local admins are removed by Alice after she becomes an admin of the world *w* and she also removes all the access rules. Alice can not remove any inherited admins of a world including the admin of *UoD*. The *UoD* admin can never be deleted by an adversary, nor can anyone add oneself in it. Hence, a complete lock-out of a world can never take place in this system. \square

Scenario 6. *A world X is created whose is-a parent is University and is-in Gujarat (is-a state contained in India). A role called “Adjunct Faculty” is defined in University which gets inherited in X. A conference called EduCon is held in Karnataka (is-a state contained in India) which defines an access rule $10000 \leftarrow AdjunctFaculty\{X, Gujarat\}$.*

Bob has Frame-level privileges on X. And he changes the is-a parent of X to “IT school”. Now, because of the is-a parent change, the role adjunct faculty (inherited from the is-a parent University) will no longer be present in X. And all the Adjunct Faculties of X will not be able to view the EduCon Conference as these credentials will fail during verification.

Scenario Handling: This is the case where Bob uses his Frame-level privileges to modify the is-a parent of world *X*. Following this change, the local admins of *X* and the worlds that have imported *X* will be notified of the is-a parent change. If the is-a parent is not restored, the rule in EduCon conference world will be quarantined. The admins or the users having Frame level privileges on *X* can restore the is-a parent in two cases: when it is an adversarial change or when it is just a mistake. If it was indeed a deliberate change, then these credentials are no longer available through world *X* and the access rule in EduCon will have to be quarantined. \square

6 Conclusions and Future Work

Personally Identifiable information, context sensitive information, confidential information, etc., all contribute to utilitarian knowledge. In a setup where sev-

eral organizations are willing to share information with one another, integrity of utilitarian data exchange becomes a concern. Within an organization, owner of the information can grant access as she knows who all may access the information. But in case of inter-organizational information sharing, the owner of the information may not be aware of the consumer of the information. There is a need of different access mechanism for such a setup. We provide MWF as a trusted framework for exchanging information across different organizations and a subsystem called credential-based access control integrated in MWF.

Proposed credential-based access control is a robust and scalable mechanism which uses credentials of a user to authorize and deny access via access rules. Credentials of a user is formed by her participation in different worlds. Her participation in a world may grant her privileges over several other worlds. Scalability is addressed with the semantics of privilege percolation in the containment hierarchy.

For the future, we plan to introduce a risk modeling framework for access rules that can evaluate the risk of adding an access rule to a given world based on the world itself as well as the credentials that it requires.

Acknowledgements. We would like to thank the reviewers of the conference ODBASE14 to provide their valuable feedback on this work.

References

1. Agarwal, S., Sprick, B., Wortmann, S.: Credential based access control for semantic web services. In: AAAI Spring Symposium-Semantic Web Services, vol. 1 (2004)
2. Barker, S.: Personalizing access control by generalizing access control. In: Proceedings of the 15th ACM Symposium on Access Control Models and Technologies, pp. 149–158. ACM (2010)
3. Bertino, E., Bonatti, P.A., Ferrari, E.: Trbac: A temporal role-based access control model. ACM Transactions on Information and System Security (TISSEC) 4(3), 191–233 (2001)
4. Bhatti, R., Bertino, E., Ghafoor, A.: A trust-based context-aware access control model for web-services. Distributed and Parallel Databases 18(1), 83–105 (2005)
5. Bhatti, R., Ghafoor, A., Bertino, E., Joshi, J.B.: X-grbac: an xml-based policy specification framework and architecture for enterprise-wide access control. ACM Transactions on Information and System Security (TISSEC) 8(2), 187–227 (2005)
6. Chen, L., Crampton, J.: Risk-aware role-based access control. In: Meadows, C., Fernandez-Gago, C. (eds.) STM 2011. LNCS, vol. 7170, pp. 140–156. Springer, Heidelberg (2012)
7. Fong, P.W.: Relationship-based access control: protection model and policy language. In: Proceedings of the First ACM Conference on Data and Application Security and Privacy, pp. 191–202. ACM (2011)
8. Joshi, J.B., Bertino, E., Latif, U., Ghafoor, A.: A generalized temporal role-based access control model. IEEE Transactions on Knowledge and Data Engineering 17(1), 4–23 (2005)
9. Laurie, B.: Access control (v0. 1) (2009)

10. Lee, A.J.: Credential-based access control. In: Encyclopedia of Cryptography and Security, pp. 271–272. Springer (2011)
11. Ni, Q., Bertino, E., Lobo, J., Brodie, C., Karat, C.M., Karat, J., Trombetta, A.: Privacy-aware role-based access control. ACM Transactions on Information and System Security (TISSEC) 13(3), 24 (2010)
12. Orlowska, E.: Kripke semantics for knowledge representation logics. Studia Logica 49(2), 255–272 (1990)
13. Sakharov, A.: Horn clause. From MathWorld—A Wolfram Web Resource, created by Eric W. Weisstein, <http://mathworld.wolfram.com/HornClause.html>
14. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-based access control models. Computer 29(2), 38–47 (1996)
15. Srinivasa, S.: Aggregating operational knowledge in community settings. In: Meersman, R., et al. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 789–796. Springer, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-33615-7_23
16. Srinivasa, S., Agrawal, S., Jog, C., Deshmukh, J.: Characterizing utilitarian aggregation of open knowledge. In: Proceedings of First ACM IKDD Conference on Data Sciences, Delhi, pp. 789–796. ACM Digital Library (March 2014), <http://dl.acm.org>
17. Vimercati, S.D.C.D., Foresti, S., Jajodia, S., Paraboschi, S., Psaila, G., Samarati, P.: Integrating trust management and access control in data-intensive web applications. ACM Transactions on the Web (TWEB) 6(2), 6 (2012)
18. Wang, T., Srivatsa, M., Liu, L.: Fine-grained access control of personal data. In: Proceedings of the 17th ACM symposium on Access Control Models and Technologies, pp. 145–156. ACM (2012)

A Model to Support Multi-Social-Network Applications

Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera

DIIES, University of Reggio Calabria

Via Graziella, Località Feo di Vito

89122 Reggio Calabria, Italy

{bucca,lax,s.nicolazzo,a.nocera}@unirc.it

Abstract. It is not uncommon that people create multiple profiles in different social networks, spreading out over them personal information. This leads to a multi-social-network scenario where different social networks cannot be viewed as monads, but are strongly correlated to each other. Building a suitable middleware on top of social networks to support internetworking applications is an important challenge, as the global view of the social network world provides very powerful knowledge and opportunities. In this paper, we do a first important step towards this goal, by defining and implementing a model aimed at generalizing concepts, actions and relationships of existing social networks.

1 Introduction

Online social networks have quickly become one of the most important communication media able to attract people and companies and to support a large number of applications in various contexts. The power of online social networks mainly derives from the fact that people are directly connected to each other, thus enabling the word-of-mouth effect (as in real social networks) on which information diffusion and people influencing is based.

In the dynamics of this phenomenon, an important role is played by membership overlap among different online social networks, thanks to which social networks are interconnected to each other. Indeed, when a user joins different networks, he plays the role of *bridge* for the information coming from one network to another. This happens because the information received in a social network can be manually posted (and thus diffused) on the other social network. Sometimes, automatic functions are enabled, such as auto posting from Twitter to Facebook or from Twitter to Facebook, so that each received tweet is posted on the Facebook's wall and vice versa.

The recent literature has highlighted that the aforementioned multiple-social-network perspective opens a lot of new problems in terms of analysis [7,3,5] but also new opportunities from the application point of view [4]. For example, consider the possibility of building the complete profile of users by merging all the information they spread out over the distinct social networks they belong to. This gives a considerable added value to market analysis and job recruitment

strategies, as membership overlap among social networks is often an expression of different traits of users personality (sometimes almost different identities).

Again, consider the field of identity management [10,14,32]: To trust user's identity or to identify fake profiles a cross check involving different social networks can be used.

From the above observations, it clearly follows that even though each single social network is an extraordinary source of knowledge, the information power of the social-network Web can be considerably increased if we see it as a huge global social network, composed of autonomous components with strong correlation and interaction.

However, despite the conceptual uniformity of the social-network universe, in terms of structure, basic mechanisms, main features, etc., each social network has in practice its own terms, resources, actions. This is a strong handicap for the design and implementation of applications enabling internetworking functions among different social networks, and, then, for the achievement of the above goal. As a consequence, building a suitable middleware on top of social networks to support internetworking applications is an important challenge, because the global view of the social network world can provide very powerful knowledge and opportunities.

In this paper, we do a first important step towards this goal, by defining and implementing a model aimed at generalizing concepts, actions and relationships of existing social networks.

This paper is organized as follows. In the next section, we describe the reference scenario. We give a formal definition of our graph-based conceptual model in Section 3. We implement the model by defining suitable mappings among concepts and social networks functionalities in Section 4. In Section 5, we show how our model can be profitably applied to some very relevant applications in the context of social network analysis. In Section 6, we survey the related work. Finally, we draw our conclusions and make some reasonings on possible future work in Section 7.

2 Reference Scenario

This section provides a description of the reference scenario. Moreover, it helps the reader to recall the main features of social networks. Such features are modeled in this paper.

An Online Social Network (OSN) provides powerful technical features to make communication among users easy. Its backbone consists of public profiles, which collect personal information and interests, and an articulated list of friends who are other users of the system. When a user joins a social network, she has to fill her own profile with descriptors, such as age, location, interests, photos and multimedia contents. Moreover, a OSN describes entities and connections among them. Entities are often individuals who are connected to each other by personal relationships, interactions, or information flows. The collection of friend nodes is not simply a list of close ties. It represents a microcosm inside the social network,

where each node can interact with the other ones. Because a friend list is visible to everyone, users can exploit it to trace friend links. A new participant can find and register a new friend using the friend lists of the other users.

Profiles and friend lists are only two key features of social networks. The third feature is a functionality allowing users to write comments about other users. These comments concerning a user profile are displayed prominently and are visible to anyone who can access that profile (i.e., to anyone who is in the friend list of the profile owner).

The three features (profiles, friends lists and comments) represent the basic structure of a social network site. Moreover, all social networks have a set of basic functionalities which are considered essential to qualify them as a social networking service. These functionalities are:

- the ability to set up and customize a personal profile by simple forms;
- an utility that allows members to comment others activities;
- a system that allows users to make a granular control of shared information (privacy settings);
- the ability to block an unwanted member in order to exclude him from the friend list;
- a personal home page which can contain personal information, notes and individual picture albums;
- the ability to own, form or be member of a group or a community within the network;
- the ability to include new “social applications”, or gadgets, which emphasize the information spreading;
- instantaneous messages;
- photo tagging tools;
- notification via SMS;
- photo and video sharing.

A *multi-social network* extends the concept of single social network by taking into account the multiplicity of social networks and their interconnections through *me* edges, by means of which a user may link other (social network) accounts she has.

3 The Conceptual Model

To model at an abstract level our multi-social-network scenario we use a graph. The set of nodes is partitioned into three disjoint sets P , R , and B , which correspond to the set of social profiles, the set of resources, and the set of bundles (which are resource containers), respectively.

An element of P models the profile of a user on a social network. It consists in a tuple $\langle \text{url}, \text{socialNetwork}, \text{screen-name}, [\text{personalInformation}], [\text{picture}] \rangle$, where url is the Web address that identifies and localizes the profile, socialNetwork is the commercial name of the social network which the profile belongs to, screen-name is the name chosen by the user who registered the profile to

appear in the home-page of the profile or when posting a resource, and, finally, personalInformation and picture are the information and the image which the user inserted as related to the profile. The two last elements of the tuple are optional (i.e., they can be null).

The set R models resources of the Web or created by users. A resource is represented by a tuple $\langle \text{url}, \text{type}, [\text{description}], [\text{date}] \rangle$, where url is the Web address to access the resource, type indicates the type of the resource content, and finally, description and date, which are optional, represent the string, inserted by the user who published the resource, describing the resource and the publishing date, respectively. For example, the most viewed video on YouTube [1] is a resource represented as $\langle \text{'https://www.youtube.com/watch?v=9bZkp7q19f0'}, \text{'video/mp4'}, \text{'PSY - GANGNAM STYLE'}, \text{'07/15/2012} \rangle$.

Our model includes the *bundle* set B . Indeed, commonly users do not handle a single resource, but most of the actions they do (e.g., publishing or sharing) involve more resources simultaneously. For example, a user can publish more photos or videos, can include a comment, and so on. In our model, we include all resources handled simultaneously by a user in a *bundle*. A bundle is represented by a tuple $\langle \text{uri}, [\text{description}], [\text{date}] \rangle$, where uri is the identifier of the bundle, description, which is optional, is the string chosen by the user to be shown with those resources and, finally, date represents the publishing date. As we will see next, we represent the inclusion of a resource into a bundle by means of *containing* edges.

In our model, relationships among profiles, resources and bundles are represented by direct edges of a graph. According to the specifics reported in Section 2, the set E of edges is partitioned into 8 disjoint sets, named F , M , Pu , S , T , Re , L , and Co .

The *follow* edge set $F \subseteq E = \{p_s, p_t \mid p_s, p_t \in P\}$ denotes that in the (source) profile p_s , it has been declared a certain type of relationship towards the (target) profile p_t . This kind of edge models different relationships. For example, on Facebook or Flickr, it models friendship, on LinkedIn, job contacts, and, on Twitter, *followers*. Observe that, typically, this kind of relationship occurs between users of the same social network, because it is presumable that a social network does not have interest in promoting links to profiles of another (competitor) social network.

The *me* edge set $M \subseteq E = \{p_s, p_t \mid p_s, p_t \in P\}$ denotes that the user with profile p_s has declared in this profile to have a second profile p_t . This edge allows a user to provide a link to its profile (typically) on a different social network or (sometimes) on the same social network (as a sort of alias).

The *publishing* edge set $Pu \subseteq E = \{p_s, b_t \mid p_s \in P, b_t \in B\}$ denotes that the user with profile p_s has published in this profile a bundle b_t . This edge models one of the typical actions a user does when enriches his/her profile by publishing resources.

The *shared* edge set $S \subseteq E = \{b_s, b_t \mid b_s, b_t \in B\}$ denotes that the bundle b_s (published by a user) is derived from an already published bundle b_t . This type of edge is used when a user shares an existing bundle. Indeed, this action is

represented by two edges: a publishing edge (as described before) and a shared edge from the new bundle to the existing one.

The *tagging* edge set $T \subseteq E = \{p_s, br_t, w \mid p_s \in P, br_t \in B \cup R \text{ and } w \text{ is a word}\}$, denotes that the user with profile p_s assigned the word w to describe a bundle or a resource br . By means of the tag mechanism, users contribute to resource labelling, which is necessary to carry out several actions on resources, such as searching or classification.

The *referencing* edge set $Re \subseteq E = \{b_s, p_t \mid b_s \in B, p_t \in P\}$ models the fact that a bundle b_s includes a reference to the profile p_t . For example, this occurs when a *tweet* includes a user account name.

The *like* edge set $L \subseteq E = \{p_s, pbr_t \mid p_s \in P, pbr_t \in B \cup R \cup P\}$ models the information that a user with the profile p_s expressed a preference/appreciation for a bundle, a resource or another user profile pbr_t .

The *containing* edge set $Co \subseteq E = \{b_s, r_t \mid b_s \in B, r_t \in R\}$ models the fact that a bundle b_s contains the resource r_t . For example, when a user publishes a photo p and includes a comment c , this action is modeled by creating a bundle b with a description c , a resource p , and finally, a *containing* edge from b to p .

After defining the conceptual model, we will show how to practically map real-life data from social networks to each component of the model, in such a way to build a data structure that can be used at application level (as we will show in Section 5).

4 Building the Model

Information necessary to build the model can be extracted from social networks via four technologies: (i) APIs provided by the social network; (ii) FOAF datasets; (iii) XFN microformat; and (iv) HTML parsing.

As for the first technology, social network APIs are a platform available for developers which allow the access to social-networks data so as to create applications on top of them. Usually, there are different kinds of APIs each providing specific services. Among them, the most commons are the REST API, the Search API and the Streaming API. Specifically, the REST APIs allow operations such as insert, update or deletion to be performed. The Search APIs, instead, are useful to query the database and, finally, the Streaming APIs are conceived for applications that need to receive real-time updates (such as, new posts or feeds).

The second possible strategy to extract information from social network relies on FOAF datasets. The FOAF project [2] focuses on the creation of a machine-readable ontology describing friendship relationships among users. FOAF data sources allow the representation of a whole social network without the need of a centralized database. In fact, by relying on this technology, it is possible to represent the information concerning a user account, along with the corresponding contacts and activities, through an RDF graph serialized as an XML document, according to the W3C RDF/XML syntax.

The third option makes use of XFN microformat. It allows for the representation of the kind of relationship existing between two user accounts. This is

```

1   {
2     "id": "1587099156",
3     "first_name": "Serena",
4     "gender": "female",
5     "last_name": "Nicolazzo",
6     "locale": "en_GB",
7     "name": "Serena_Nicolazzo",
8     "username": "serena.nicolazzo"
9   }

```

Listing 1.1. An example of the output of the Facebook Graph API.

obtained by empowering the set of values that the `rel` attribute of the HTML tag `<a>` (which represents a link) can assume. In our case, we focus on the value “me” (`rel='me'`) which indicates that the corresponding link represents a `me` edge.

The last data extraction strategy leverages on HMTL parsing. Processing HTML to obtain social data is the most intricate procedure. Parsing requires much time because it needs to analyze all context information from the page source code. It is a low-level way of dealing with social data. Because the code written depends on the HTML page structure, it is not stable (due to the frequent graphical changes). For this reason, this strategy needs continue maintenances. However, it remains a valid alternative when other more practical solutions (like APIs, for instance) are not available.

Now we will show some significant examples on how the information represented by our model are extracted from social networks.

As for the user profile P described in Section 3, it consists in a tuple $\langle \text{url}, \text{socialNetwork}, \text{screen-name}, \text{personalInformation}, \text{picture} \rangle$. For example, to extract the information to build the profile of a Facebook user, we use the **Graph APIs**, accessible through the url `http://graph.facebook.com/{user-id}` or `http://graph.facebook.com/{screen-name}`. The output of this API is a JSON file (see, for instance, Listing 1.1).

We can extract from this JSON the user id, her username (which correspond to our notion of screen-name) and her personal information like: first name, last name, gender, locale (chosen language). The field `url` available in our social profile object can be obtained as `http://www.facebook.com/{screen-name}`, whereas the field `picture` can be obtained by another call to the **Graph APIs**, specifically by accessing the url `http://graph.facebook.com/{user-id}/picture`.

Many social networks are equipped also with FOAF datasets. As an example, we show how the edges belonging to the set `follow` can be obtained for the social networks LiveJournal and Advogato. The FOAF datasets for both social networks are reachable through the specific URLs `http://{screen-name}.livejournal.com/data/foaf` (for LiveJournal) and `http://www.advogato.org/person/{screen-name}/foaf.rdf` (for Advogato).

An example of an XML serialization of a FOAF document is shown in Listing 1.2. In this document, the information needed to build an edge of the set `follow`

```

1 <?xml version='1.0'?>
2 <rdf:RDF
3   xmlns:lang="en"
4   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6   xmlns:foaf="http://xmlns.com/foaf/0.1/"
7   xmlns:yae="http://blogs.yandex.ru/schema/foaf/"
8   xmlns:lj="http://www.livejournal.org/rss/lj1.0/"
9   xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
10  xmlns:dce="http://purl.org/dc/elements/1.1/">
11  <foaf:Person>
12    <foaf:nick>antoninonocera</foaf:nick>
13    <foaf:name>real_name</foaf:name>
14    <foaf:openid rdf:resource="http://antoninonocera.livejournal.com"/>
15    <foaf:weblog rdf:resource="http://antoninonocera.livejournal.com"/>
16    <foaf:homepage rdf:resource="*****" dc:title="" />
17    ...
18    <foaf:knows>
19      <foaf:Person>
20        <foaf:nick>contact1</foaf:nick>
21        <foaf:member_name>contact_real_name</foaf:member_name>
22        <foaf:tagLine></foaf:tagLine>
23        <foaf:image href="http://l-userpic.livejournal.com/****/****"></foaf:image>
24        <rdfs:seeAlso rdf:resource="http://contact1.livejournal.com/data/foaf"/>
25        <foaf:weblog rdf:resource="http://contact1.livejournal.com"/>
26      </foaf:Person>
27    </foaf:knows>
28    ...
29  </foaf:Person>
30 </rdf:RDF>

```

Listing 1.2. An XML-serialized FOAF document.

```

1   <a class="OLa_url_Xvc" href="http://www.youtube.com/channel/<screen-name>">
2     rel="me" target="_blank" title="<screen-name>><Link_name></a>

```

Listing 1.3. An example of a *me* edge using XFN.

can be extracted from lines 11 to 29. Specifically, the element $\langle \text{foaf} : \text{Person} \rangle$ indicates the beginning of the portion of the document where information about a user, her contacts and, often, her activities are reported. The information about each contact is encoded as a $\langle \text{foaf} : \text{Person} \rangle$ nested inside a tag $\langle \text{foaf} : \text{knows} \rangle$.

Concerning the information about *me* edges, it can often be extracted through the XFN microformat. Some examples of social networks adopting this standard to represent *me* edges are About.me, Advogato, Boards, Facebook, Flickr, Google+, and Twitter. Listing 1.3 shows the code representing a *me* edge in the social network Google+. The code at line 2 represents the explicit declaration that the corresponding link encodes a relation of type *me*.

Another interesting example regards the extraction of the information needed to build a *publishing* edge. Consider the social network LinkedIn. It provides a search API, called Job Lookup API, to obtain information about jobs that can be accessed at the address [http://api.linkedin.com/v1/jobs/job_id:\(id,company,posting-date\)](http://api.linkedin.com/v1/jobs/job_id:(id,company,posting-date)). The XML output produced by the call to this API is reported in Listing 1.4. In this case, when a company proposes a new job position (*publishing*), we model this event by adding two objects: (i) a bundle and (ii) a *publishing* edge between the profile of the company and the bundle just created (see Section 3). As for the bundle, the field *uri* is mapped to the element

```

1   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2   <job>
3       <id>1511685</id>
4       <expiration-timestamp>1304030488000</expiration-timestamp>
5       <company>
6           <id>2299433</id>
7           <name>Cloudera</name>
8       </company>
9       <position>
10      <title>Technical Writer</title>
11      <location>
12          <name>San Francisco Bay Area</name>
13          <country>
14              <code>us</code>
15          </country>
16          <location>
17      </position>
18      <location-description>San Francisco or Palo Alto, CA</location-description>
19      <job-poster>
20          <id>hQ4ruu3J2q</id>
21          <first-name>Paul</first-name>
22          <last-name>Battaglia</last-name>
23          <headline>Technical Writer at Cloudera</headline>
24      </job-poster>
25  </job>

```

Listing 1.4. An example of the output of the LinkedIn Job Lookup API.

< *id* > (line 3), whereas the field **description** is obtained from the elements <**company**> (lines 5-6), <**position**> (lines 9-17), and <**localition-description**> (line 18). The *publishing* edge is associated with the user profile whose identifier is specified by the element <**job-poster**> (lines 19-24) and has the new created bundle as target.

Now, consider the case of the publishing of a new tweet containing a resource and referencing another user in the Twitter social network. Our model represents this action by adding the following objects: (i) a bundle, (ii) a resource, (iii) a *publishing* edge, (iv) a *containing* edge, and (v) a *referencing* edge. In this case, all information required by our model is extracted from Twitter by means of the method *GET statuses/user_timeline* of the Twitter APIs. Listing 1.5 shows an example of the output of this API. Specifically, line 6 is the tweet identifier and is mapped to the field **uri** of the bundle. The bundle field **description** is obtained by suitably parsing lines 24-30. The bundle is linked to the publisher user by means of a *publishing* edge. As mentioned above, a new resource is added and associated with the bundle by means of a *containing* edge. Information needed to create the resource is extracted from lines 11-23. In particular, the field **url** is obtained from line 15, the field **type** from line 20, and, finally, the field **description** is extracted from lines 19, 20 and 21. The tweet in the example references another user and this action is modeled by adding a *referencing* from the bundle to the user specified on lines 9 and 10.

An important feature, common to almost all social networks, is the possibility to appreciate a resource or another user profile. In our model, this concept is represented by means of the *like* edge. Consider the social network About.me, in which a user is allowed to favor another user profile, thus making an “endorsement”. Information about this action is obtained by calling the method http://api.about.me/api/v2/json/user/view/<screen_name>

```

1 {
2   "coordinates": null,
3   "favorited": false,
4   "truncated": false,
5   "created_at": "Wed_Aug_29_17:12:58+0000+2012",
6   "id_str": "240859602684612608",
7   "entities": {
8     "hashtags": [ ],
9     "user_mentions": [{"indices": [3, 10], "id_str": "<user_id>", "screen_name": "<user_screen_name>" },
10    {"name": "<user_real_name>", "id": <user_id>}]
11    "media": [ {
12      "id": 266031293949698048,
13      "id_str": "266031293949698048",
14      "indices": [17, 37],
15      "media_url": "http://pbs.twimg.com/media/A7EiDWcCYAAZT1D.jpg",
16      "media_url_https": "https://pbs.twimg.com/media/A7EiDWcCYAAZT1D.jpg",
17      "url": "http://t.co/bAJE6Vom",
18      "display_url": "pic.twitter.com/bAJE6Vom",
19      "expanded_url": "http://twitter.com/BarackObama/status/266031293945503744/photo/1",
20      "type": "photo",
21      "sizes": {...}
22    }]
23  },
24  "in_reply_to_user_id_str": null,
25  "contributors": null,
26  "text": "Introducing the Twitter Certified Products Program: https://t.co/MjJ8xAnT",
27  "retweet_count": 121,
28  "in_reply_to_status_id_str": null,
29  "id": 240859602684612608,
30  "geo": null,
31  "retweeted": false,
32  "possibly_sensitive": false,
33  "in_reply_to_user_id": null,
34  "place": null,
35  "user": {...},
36  "in_reply_to_screen_name": null,
37  "source": "<a href=\"http://sites.google.com/site/yorufukurou/\" rel=\"nofollow\">YoruFukurou</a>",
38  "in_reply_to_status_id": null
39 }

```

Listing 1.5. An example of the output of the Twitter API method *user_timeline*.

of the About.me API. Listing 1.6 reports an example of the output of this method. The returned information has to be seen from the “caller” point of view (i.e., the authenticated user), therefore the line 21 indicates that the user *screen_name* is in the favourite list of the authenticated user. According to our model, a *like* edge from the authenticated user to the user with the given *screen_name* is created. A similar reasoning can be applied also for “g+1” of Google+ and “Like” of Facebook.

So far, we have seen how to extract information from different social networks and how to map them to the concepts defined in our model. Once this mapping has been done, a data-structure is obtained. It can be serialized using the XML language. In the following, we will show some details about the XML schema underlying our model.

Figures 1, 2 and 3 show the tree-based representation of our XML schema. The root element is *SocialGraph* and contains two unbounded sets of elements, namely *SocialNode* and *SocialEdge*. An element *SocialNode* is specialized in one of the following complex types: *SocialProfile*, *Resource*, or *Bundle*. The element *SocialEdge* is specialized in one of the following complex types: *Follow*, *Me*, *Publishing*, *Tagging*, *Shared*, or *Referencing*. Each complex-type in this XML is defined according to the corresponding objects defined in Section 3. Listing

```

1{
2    "status": 200,
3    "profile": "http://about.me/<screen_name>",
4    "user_name": "test_account",
5    "first_name": "test22",
6    "last_name": "tester",
7    "display_name": "test22_tester",
8    "header": "my_headline",
9    "bio": "test_this_is_one!!!!",
10   "background": "http://about.me/.../<screen_name>_1326415784_79.jpg",
11   "mobile_background": "",
12   "email_searchable": true,
13   "email_public": false,
14   "avatar": "http://about.me/.../<screen_name>_1325746595_83.jpg",
15   "img_base_url": "http://about.me/.../thumbnail",
16   "thumbnail_291x187": "http://about.me/.../291x187/<screen_name>.jpg",
17   "thumbnail1": "http://about.me/.../803x408/<screen_name>.jpg",
18   "thumbnail12": "http://about.me/.../260x176/<screen_name>.jpg"
19   "thumbnail13": "http://about.me/.../198x134/<screen_name>.jpg",
20   "thumbnail14": "http://about.me/.../161x109/<screen_name>.jpg",
21   "is_fav": true
22}

```

Listing 1.6. An example of the output of the About.me API method *view*.

1.7 reports a fragment of the XML schema whose tree-representation has been described above.

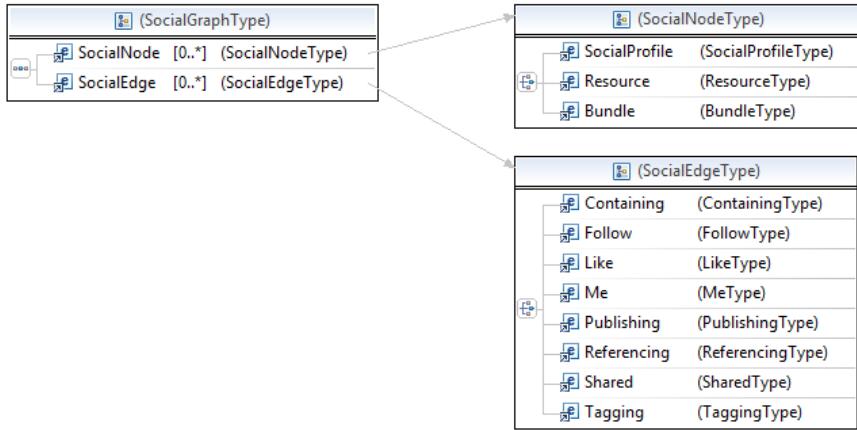


Fig. 1. A portion of the tree representation of our XML Schema (1/3)

We conclude this section by showing in Listing 1.8 an example of a fragment of an XML document derived from the XML Schema described above. Lines 6-20 show the definition of the Twitter profiles of two of the authors of this paper. In lines 39-44, a *follow* edge among them is defined. Lines 22-37 represent the definition of a new bundle and a new resource of type “photo” (line 33). The bundle is published by one of the authors of this paper and the *publishing* action

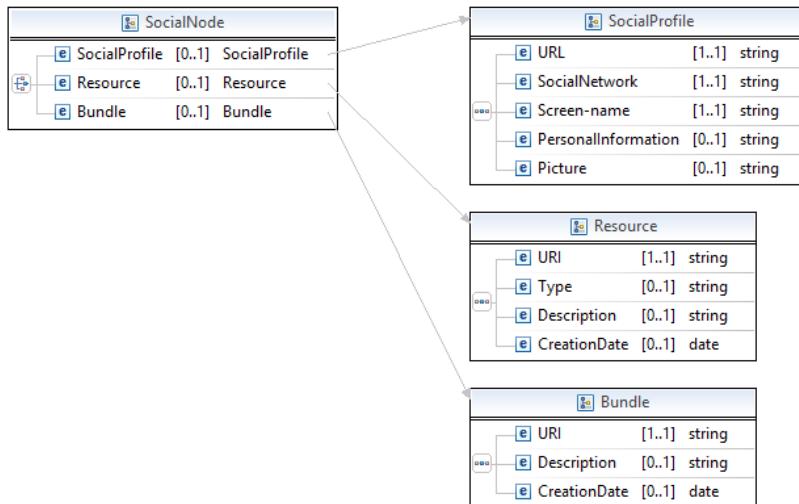


Fig. 2. A portion of the tree representation of our XML Schema (2/3)

is encoded in lines 46-51. The resource is contained in the bundle as shown in lines 53-58 with the definition of a *containing* edge. Finally, the information that one of the authors has two accounts in Twitter and Facebook is modeled in lines 60-74.

5 Case Study

In this section, we briefly describe how our model has been profitably applied to two applications very relevant in the context of social network analysis. The first application regards the extraction of information from a multi-social-network scenario, the second one concerns a particular analysis done on social network data.

It is well known that any analysis activity on social network users needs a preliminary task implementing the extraction of data from social networks. In the past, several visit strategies have been adopted, such as Breadth First Search [31], Random Walk [20] or Metropolis- Hastings Random Walk [27]. In all these cases, data analysis focused on a single social network and data extraction was a quite simple task because there was not the problem of receiving data from different sources.

When data extraction involves different social networks, having a model that is able to handle indifferently data from different social networks is a very useful tool.

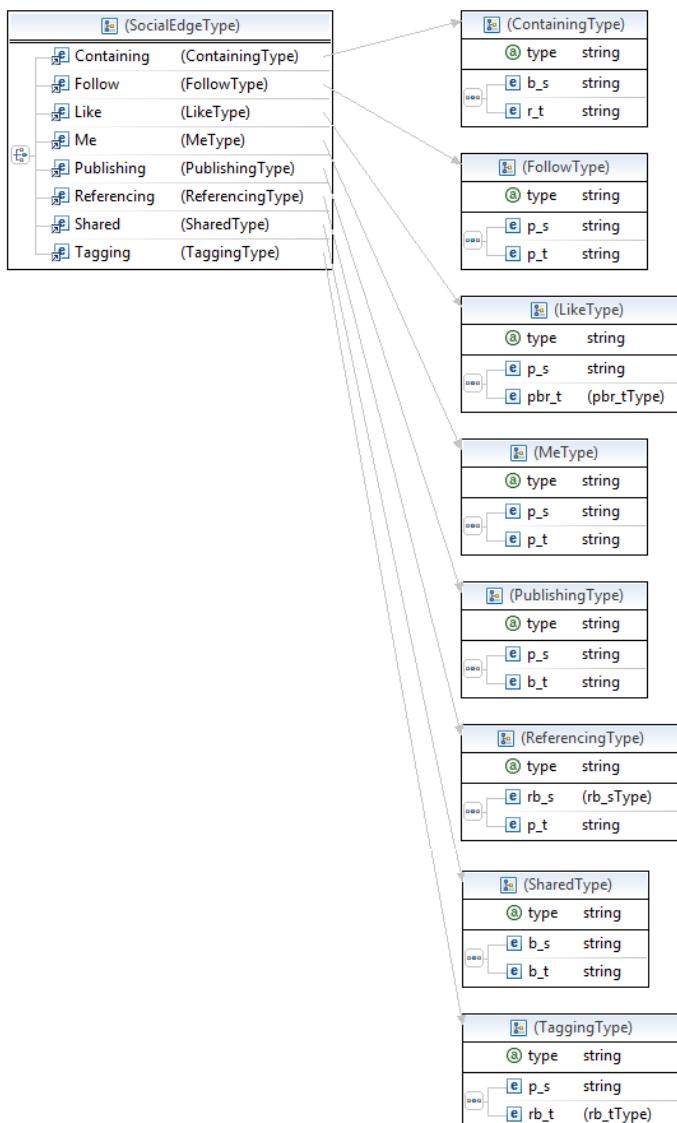


Fig. 3. A portion of the tree representation of our XML Schema (3/3)

The model defined here has been successfully used in the SNAKE system [8], a tool supporting the extraction of data from social network accounts.

```

1 ...
2     <element name="SocialGraph">
3         <complexType>
4             <sequence>
5                 <element ref="tns:SocialNode" minOccurs="0" maxOccurs="unbounded" />
6                 <element ref="tns:SocialEdge" minOccurs="0" maxOccurs="unbounded" />
7             </sequence>
8         </complexType>
9     </element>
10
11    <element name="SocialNode">
12        <complexType>
13            <choice>
14                <element ref="tns:SocialProfile" />
15                <element ref="tns:Resource" />
16                <element ref="tns:Bundle" />
17            </choice>
18        </complexType>
19    </element>
20
21    <element name="SocialProfile">
22        <complexType>
23            <sequence>
24                <element name="URL" type="string" minOccurs="1" maxOccurs="1" />
25                <element name="SocialNetwork" type="string" minOccurs="0"
26                         maxOccurs="1" />
27                <element name="Screen-name" type="string" minOccurs="1"
28                         maxOccurs="1" />
29                <element name="PersonalInformation" type="string" minOccurs="0"
30                         maxOccurs="1" />
31                <element name="Picture" type="string" minOccurs="0"
32                         maxOccurs="1" />
33            </sequence>
34        </complexType>
35        <xss:key name="spkey">
36            <xss:selector xpath="SocialGraph/SocialNode/SocialProfile" />
37            <xss:field xpath="@URL" />
38        </xss:key>
39    </element>
40
41...
42
43    <element name="SocialEdge">
44        <complexType>
45            <choice>
46                <element ref="tns:Containing" />
47                <element ref="tns:Follow" />
48                <element ref="tns:Like" />
49                <element ref="tns:Me" />
50                <element ref="tns:Publishing" />
51                <element ref="tns:Referencing" />
52                <element ref="tns:Shared" />
53                <element ref="tns:Tagging" />
54            </choice>
55        </complexType>
56    </element>
57 ...

```

Listing 1.7. A portion of our XML Schema.

The second application that benefits from our model concerns the problem of identifying users on the Web. This problem has received a great attention in several application scenarios, such as personalization [10,14,32].

A common approach to address this problem utilizes profile matching techniques typically based on a set of identification properties, such as username, to find user corresponding identity. In [6], an improvement of this approach is proposed. In particular, a new notion of profile similarity is defined, by combining a string similarity between the associated usernames with a contribution based on a suitable recursive notion of common-neighbor similarity. The computation of the second contribution requires to compare profiles coming from different social networks, which could be quite heterogeneous.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <nssocialGraph xmlns:nss="http://www.example.org/SocialGraph"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.example.org/SocialGraph.xsd">
5
6   <nssocialNode>
7     <nssocialProfile>
8       <nssURL>twitter.com/antoninonocera</nssURL>
9       <nssocialNetwork>Twitter</nssocialNetwork>
10      <nsscreenName>AntoninoNocera</nsscreenName>
11    </nssocialProfile>
12  </nssocialNode>
13
14  <nssocialNode>
15    <nssocialProfile>
16      <nssURL>twitter.com/serenanicolazzo</nssURL>
17      <nssocialNetwork>Twitter</nssocialNetwork>
18      <nsscreenName>serenanicolazzo</nsscreenName>
19    </nssocialProfile>
20  </nssocialNode>
21
22  <nssocialNode>
23    <nssBundle>
24      <nssURI>240859602684612608</nssURI>
25      <nssDescription>Take a look to this beautiful photo :-)</nssDescription>
26      <nssCreationDate>2012-08-29</nssCreationDate>
27    </nssBundle>
28  </nssocialNode>
29
30  <nssocialNode>
31    <nssResource>
32      <nssURI>http://pbs.twimg.com/media/A7EiDWcCYAAZTiD.jpgI</nssURI>
33      <nssType>photo</nssType>
34      <nssDescription>A pic of us</nssDescription>
35      <nssCreationDate>2012-08-29</nssCreationDate>
36    </nssResource>
37  </nssocialNode>
38
39  <nssocialEdge>
40    <nssFollow type="F">
41      <nssp_s>twitter.com/antoninonocera</nssp_s>
42      <nssp_t>twitter.com/serenanicolazzo</nssp_t>
43    </nssFollow>
44  </nssocialEdge>
45
46  <nssocialEdge>
47    <nssPublishing type="Pu">
48      <nssp_s>twitter.com/antoninonocera</nssp_s>
49      <nssb_t>240859602684612608</nssb_t>
50    </nssPublishing>
51  </nssocialEdge>
52
53  <nssocialEdge>
54    <nssContaining type="Co">
55      <nssb_s>240859602684612608</nssb_s>
56      <nssr_t>http://pbs.twimg.com/media/A7EiDWcCYAAZTiD.jpg</nssr_t>
57    </nssContaining>
58  </nssocialEdge>
59
60  <nssocialNode>
61    <nssocialProfile>
62      <nssURL>www.facebook.com/antonino.nocera.35</nssURL>
63      <nssocialNetwork>Facebook</nssocialNetwork>
64      <nsscreenName>Antonino Nocera</nsscreenName>
65      <nssPersonalInformation>Male</nssPersonalInformation>
66    </nssocialProfile>
67  </nssocialNode>
68
69  <nssocialEdge>
70    <nssMe>
71      <nssp_s>twitter.com/antoninonocera</nssp_s>
72      <nssp_t>www.facebook.com/antonino.nocera.35</nssp_t>
73    </nssMe>
74  </nssocialEdge>
75 ...
76 </nssocialGraph>
```

Listing 1.8. An example of an XML document.

It is worth noting that the papers referred above (i.e., [8] and [6]) are completely not focused on the modelling (and implementation) aspects faced in this paper, so that this paper adds original relevant work.

6 Related Work

Over time, social network analysts have mainly used two kinds of tools from mathematics to represent information about patterns of ties among social actors: graphs and matrices.

Examples of the first group are: Kronecker graphs model [19]; the class of model networks presented in [22] that are generalizations of the much-studied random graph of Erdős and Rényi [11] to model social networks; the multiplicative attribute graph model presented in [17] that considers information about properties of the nodes of the network; and others, such as [9,29,28], that model their application scenarios with graphs.

The approaches that adopt matrices representation to model social networks [18,26] belong to the second group. Specifically, the approach of [18] incorporates social influence processes in the specification of a weight matrix W , whereas the approach of [26] uses a tensor to model the interaction between resources and users.

A minor trend is to formalize social networks through a three phase model [24,13]. This model was developed in [30] to identify critical social networks activities.

The hypergraph theory [16] allows a hyperedge to connect an arbitrary number of vertices instead of two in regular graphs. For instance, Ghoshal et al. [12] introduce a random hypergraph model to describe the ternary relationship among one user, one resource and one tag, thus making the model more flexible in the representation of many peculiar properties of folksonomies. In [33], the authors propose an hypergraph model to illustrate the emergence of some statistical properties in a folksonomy such as: degree distribution, clustering coefficients and average distance between nodes.

Another worthy-of-mention category of approaches adopt suitable models with the purpose of creating global user profiles by means of deep analyses of their behavior accessing multiple social networks. Often, the application scenarios of these approaches are those of ontologies and folksonomies.

An interesting approach, in this context, is TAGMAS (Tag MAnagement System) [15]. It relies on an ontology to uniformly describing tags and tagging actions in several distinct folksonomies.

The problem of integrating data of different social sites is addressed in [25] and [23]. The basic idea of [25] is to collect data from different folksonomies and, then, to create groups of tags by adopting suitable clustering algorithms. After this step, these groups are mapped onto the concepts of an ontology. In [23], the authors propose an approach which gathers data about user activities on social sites. Suitable ontologies are used both to analyze these data and model user interests. An important approach belonging to this category is that of [21]. The author formulates an abstract model of semantic-social networks, in the form of a tripartite graph of persons, concepts and instances. Hence, incorporating actors in this model, he extends the traditional concept of ontologies (composed by concepts and instances). By the way, because the referring scenario of this paper is that of folksonomies, this model represents only one action (i.e., *tagging*). It is

defined as a ternary association between user, concept and object. More in detail, the set of shared object and the set of keywords defined by users themselves are extracted from social networks. These collections are, then, used to obtain the emergence of a community-based ontologies.

Even if our approach has some common aspects with these proposals, none of them consider the possibility of integrating information coming from different and heterogeneous social networks. This additional feature of our model makes it strongly different from the approaches presented here, because the uniform representation of all the peculiarities of different social networks is a non-trivial task and needs ad-hoc solutions to be pursued.

7 Conclusion

In this paper, we have defined and implemented a model aimed at creating a middleware on top of existing online social networks. The goal is to provide a (conceptual) layer able to facilitate design and implementation of applications relying on the internetworking nature of online social networks. As a matter of fact, the multiplicity of social networks together with users' membership overlap, result in a multiplicative effect in terms of information power. Indeed, correlation, integration, negotiation of information coming from different social networks offer a lot of strategic knowledge whose benefits are still unexplored.

The approach followed in this paper was mainly practical, in the sense that we solved the trade-off between complexity/expressiveness of the conceptual model and implementation issues in favor of the latter. In other words, despite a somewhat naive model focused on a few crucial concepts underlying real-life social networks, the resulting benefits from the implementation perspective appears considerable. This is shown in the paper by means of case study, but we believe that a lot of further applications may confirm the relevance of our approach. We plan to show this in our future work.

Acknowledgment. This work has been partially supported by the TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research and by the Program “Programma Operativo Nazionale Ricerca e Competitività” 2007-2013, project BA2Kno (Business Analytics to Know) PON03PE_00001_1, in “Laboratorio in Rete di Service Innovation”, funded by the Italian Ministry of Education, University and Research.

References

1. Videotrue. The most viewed videos on Youtube in the World of all time (2014), <http://en.videotrue.com/all/youtube/all-time>
2. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.91. Technical report, Tech. rep. ILRT Bristol (2000), <http://xmlns.com/foaf/spec/20071002.html>
3. Buccafurri, F., Foti, V., Lax, G., Nocera, A., Ursino, D.: Bridge Analysis in a Social Internetworking Scenario. *Information Sciences* 224, 1–18 (2013)

4. Buccafurri, F., Lax, G., Nicolazzo, S., Nocera, A., Ursino, D.: Driving Global Team Formation in Social Networks to Obtain Diversity. In: Casteleyn, S., Rossi, G., Winckler, M. (eds.) ICWE 2014. LNCS, vol. 8541, pp. 410–419. Springer, Heidelberg (2014)
5. Buccafurri, F., Lax, G., Nocera, A., Ursino, D.: Crawling social internetworking systems. In: Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), pp. 506–510. IEEE Computer Society (2012)
6. Buccafurri, F., Lax, G., Nocera, A., Ursino, D.: Discovering Links among Social Networks. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 467–482. Springer, Heidelberg (2012)
7. Buccafurri, F., Lax, G., Nocera, A., Ursino, D.: Moving from social networks to social internetworking scenarios: The crawling perspective. *Information Sciences* 256, 126–137 (2014)
8. Buccafurri, F., Lax, G., Nocera, A., Ursino, D.: A system for extracting structural information from social network accounts. *Software: Practice and Experience* (2014), doi:10.1002/spe.2280
9. Caldarelli, G.: Scale-Free Networks: Complex Webs in Nature and Technology. Number 9780199211517 in OUP Catalogue. Oxford University Press (2007)
10. Carmagnola, F., Cena, F.: User identification for cross-system personalisation. *Information Sciences* 179(1-2), 16–32 (2009)
11. Erdős, P., Rényi, A.: On Random Graphs, I. *Publicationes Mathematicae* 6, 290–297 (1959)
12. Ghoshal, G., Zlatić, V., Caldarelli, G., Newman, M.E.J.: Random hypergraphs and their applications. *Physical Review E* 79(6), 066118 (2009)
13. Greve, A., Salaff, J.W.: Social networks and entrepreneurship. *Entrepreneurship Theory and Practice* 28(1), 1–22 (2003)
14. Iofciu, T., Fankhauser, P., Abel, F., Bischoff, K.: Identifying users across social tagging systems. In: Proc. of the International Conference on Weblogs and Social Media (ICWSM 2011), Barcelona, Catalonia, Spain. The AAAI Press (2011)
15. Iturrioz, J., Diaz, O., Arellano, C.: Towards federated web2.0 sites: The tagmas approach. In: Tagging and Metadata for Social Information Organization Workshop, WWW 2007 (2007)
16. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: applications in VLSI domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 7(1), 69–79 (1999)
17. Kim, M., Leskovec, J.: Modeling social networks with node attributes using the multiplicative attribute graph model. arXiv preprint arXiv:1106.5053 (2011)
18. Leenders, R.T.: Modeling social influence through network autocorrelation: constructing the weight matrix. *Social Networks* 24(1), 21–47 (2002)
19. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research* 11, 985–1042 (2010)
20. Lovász, L.: Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty* 2(1), 1–46 (1993)
21. Mika, P.: Ontologies are us: A unified model of social networks and semantics. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 522–536. Springer, Heidelberg (2005)
22. Newman, M.E.J., Watts, D.J., Strogatz, S.H.: Random graph models of social networks. *Proceedings of the National Academy of Sciences* 99(suppl. 1), 2566–2572 (2002)

23. Noor, S., Martinez, K.: Using social data as context for making recommendations: an ontology based approach. In: Proceedings of the 1st Workshop on Context, Information and Ontologies, p. 7. ACM (2009)
24. Romm, C., Pliskin, N., Clarke, R.: Virtual communities and society: toward an integrative three phase model. International Journal of Information Management 17(4), 261–270 (1997)
25. Specia, L., Motta, E.: Integrating folksonomies with the semantic web. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 624–639. Springer, Heidelberg (2007)
26. Stewart, A., Diaz-Aviles, E., Nejdl, W., Marinho, L.B., Nanopoulos, A., Schmidt-Thieme, L.: Cross-tagging for personalized open social networking. In: Proceedings of the 20th ACM Conference on Hypertext and Hypermedia, pp. 271–278. ACM (2009)
27. Stutzback, D., Rejaie, R., Duffield, N., Sen, S., Willinger, W.: On unbiased sampling for unstructured peer-to-peer networks. In: Proc. of the International Conference on Internet Measurements, Rio De Janeiro, Brasil, pp. 27–40. ACM (2006)
28. Wang, A.H.: Don't follow me: Spam detection in twitter. In: Proceedings of the 2010 International Conference on Security and Cryptography (SECRYPT), pp. 1–10. IEEE (2010)
29. Wang, X., Wei, F., Liu, X., Zhou, M., Zhang, M.: Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 1031–1040. ACM (2011)
30. Wilken, P.H.: Entrepreneurship: A comparative and historical study. Ablex, Norwood (1979)
31. Ye, S., Lang, J., Wu, F.: Crawling online social graphs. In: Proc. of the International Asia-Pacific Web Conference (APWeb 2010), Busan, Korea, pp. 236–242. IEEE (2010)
32. Zafarani, R., Liu, H.: Connecting corresponding identities across communities. In: Proc. of the International Conference on Weblogs and Social Media (ICWSM 2009), San Jose, CA, USA. The AAAI Press (2009)
33. Zhang, Z., Liu, C.: A hypergraph model of social tagging networks. Journal of Statistical Mechanics: Theory and Experiment 2010(10), P10005 (2010)

A Method for Detecting Behavior-Based User Profiles in Collaborative Ontology Engineering

Sven Van Laere, Ronald Buyl, and Marc Nyssen

Department of Biostatistics and Medical Informatics - Public Health,
Vrije Universiteit Brussel, Laarbeeklaan 103, 1090 Jette, Belgium
`{svvlaere,rbuyl,mnyssen}@vub.ac.be`

Abstract. Ontology engineering is far from trivial and most collaborative methods and tools start from a predefined set of rules, stakeholders can have in the ontology engineering process. We, however, believe that the different types of user behavior are not known *a priori* and depend on the ontology engineering project. The detection of such user profiles based on unsupervised learning allows finding roles and responsibilities along peers in a collaborative setting. In this paper, we present a method for automatic detection of user profiles in a collaborative ontology engineering environment by means of the K-means clustering algorithm only by looking at the type of interactions a user makes. In this paper we use the GOSPL ontology engineering tool and method to demonstrate this method. The data used to demonstrate the method stems from two ontology engineering projects involving respectively 42 and 36 users.

Keywords: Collaborative Ontology Engineering, User Profiling, clustering.

1 Introduction

An ontology is commonly defined as: “*a [formal,] explicit specification of a [shared] conceptualization*” [11], and is key in enabling semantic interoperability between autonomously developed information systems belonging to a community of stakeholders. The construction of ontologies are far from trivial and require methods and tools to support these communities in the ontology engineering process.

Depending on the method and tool selected for collaborative ontology engineering, different predefined roles are proposed. However, we believe that the different “types” of users (or user behaviors) are *a priori* not known, and that the predefined roles and responsibilities should at least be complemented with types based on behavior. When a project starts, roles are assigned based on the confidence and reliability a project leader has in a person. Behavior-based approaches use the user’s behavior as a model, commonly relying on machine-learning techniques to discover useful patterns in it [22].

Often task groups are assigned in an ontology engineering project where the project leader attributes tasks among different team members. However, for the

project leader it is difficult to label users adequately and more importantly he does not know how users prefer to interact with a system/tool. An important assumption in this paper is thus: *the types of users in a community or communities are not known beforehand*. The potential of identifying many different types are vast, including: (i) personalizing the interaction a user has with a system to render their tasks more efficient (adaptive hypermedia), (ii) detect a group of users' expertise and thus provide means to fully exploit that group's capabilities, (iii) composing tasks groups with a variety of skills and competencies to automate some of the ontology engineering processes.

This paper presents some preliminary results in answering the following research question: "How can we discover user types by analysing the interactions between users, and between users in a collaborative ontology engineering environment?" This paper is organized as follows: Section 2 mentions the applications of user profiling and their application in collaborative ontology engineering; Section 3 provides our approach to annotate the data of an existing collaborative ontology engineering tool to demonstrate the extraction of social interactions; Section 4 gives an overview of the method for user profiling two ontology engineering datasets; Section 5 applies the user profiling algorithm on two ontology engineering datasets and gives an interpretation of the found profiles; In Section 6 we discuss the outcome of the clustering algorithm and conclude the paper.

2 Related Work

User profiling is an emerging research field that considers the application of finding structures in the way people behave. If we observe online communities one can derive a lot of information from the social interactions between users, and between a user and a system. People can be classified according to behavior, taste, effort, etc. Related work on user profiling can be found in different fields: e-commerce [20], [29], computer-supported cooperative work (CSCW) [21], [25], web browsing [18], [19], [30], news feeds [2], [17], [27], etc.

Also in the field of information governance, user profiling is emerging. Gartner defines information governance as: "*the specification of decision rights and an accountability framework to encourage desirable behavior in the valuation, creation, storage, use, archival and deletion of information. It includes the processes [actions], roles [actors], standards and metrics [actands] that ensure the effective and efficient use of information in enabling an organization to achieve its goals.*"¹ Most scientific papers are directly inspired by traditional data quality management and IT governance [15], and propose deterministic role patterns and decision domains with a predefined terminology. Yet, it is necessary that these models need to be flexible at run-time, i.e. *contingent* upon issues [6]. Here, our approach can help by making the process more generic.

When we investigate these topics it is key to have a clear understanding of what roles are, how they relate to human behaviors, and how these behaviors

¹ http://blogs.gartner.com/debra_logan/2010/01/11/ (last accessed on August 30, 2014).

can be captured in terms of online community features. A discussion about the definition of a role can be found in [10]. In their discussion they state that a user role can arise either from the social context of a person and the dynamics of his/her relationships or from repeated interactions and agreements across practices. In this work we adopt the second definition of role. Usually a set of behavioral dimensions is used to distinguish user profiles; here we use types of interactions. Examples of roles mentioned in the literature are: *newbies*, *experts* or *lurkers* [28].

Each of these roles is identified by a set of behaviors, (or behavioral dimensions), such as engagement, contribution, popularity, participation, etc. The general procedure to model behavior in an online community is by translating them into measurable behavioral features from the social network graph with an associated intensity level (see e.g. [13], [23]). *In contrast we will use features based on the kind of interactions users prefer.*

This work focusses on applying a data mining method for extracting user profiles in ontology engineering. Though not much related can be found in the field of ontology engineering, [6] aimed at relating performance indicators with user type, and therefore applied simple statistical measures for classifying users. [9] described how they sought and discovered collaboration patterns in the creation of ontologies in the medical domain. Indirectly, those collaboration patterns give insights about the types of users in that experiment. The identification of these roles will then later on be used to target our objectives and make work in the ontology engineering process lighter.

3 Social Interactions in Ontology Engineering

3.1 Adopting a Collaborative Ontology Engineering Method

There exist quite a few collaborative ontology engineering methods: HCOME [16], DILIGENT [24], Business Semantics Management [5], etc. The method adopted in this paper is GOSPL², which has been described in [8]. GOSPL was chosen for its explicit social component (communities were promoted to first class citizen and the use of natural language definitions – called *glosses* – of concepts in the community’s communication are not subordinate to the formal descriptions of concepts), and more importantly the availability of a data set from an experiment described later in this paper.

In GOSPL, concepts are both represented formally and informally. Formally by means of *lexons* and informally by means of *glosses*. An example of a lexon is <Cultural Domain, Concert, is a, subsumes, Event>, which states that, in the “Cultural Domain” community, the concept referred to with term “Concerts” plays the role of “is a” on the concept with term “Event”, and the concept with term “Event” plays the role of “subsumes” on the concept with term “Concert”.

² GOSPL stands for “Grounding Ontologies with Social Processes and natural Language”.

A lexon should ideally result in two meaningful³ sentences when read in both directions. But assuring this quality is the responsibility of the community.

Synonyms are agreements that two terms refer to the same concept, and *gloss-equivalences* are agreements that two descriptions refer to the same concept. Ideally, if two communities agree that two descriptions refer to the same concept, the labels associated with those descriptions should be considered equal as well; this is called the glossary consistency principle [7].

Fig. 1 depicts the different processes in GOSPL. Starting from co-evolving communities and requirements, the informal descriptions of key terms have to be gathered before formally describing those concepts. Communities define the semantic interoperability requirements, out of which a set of key terms is identified. Those terms need to be informally described before the formal description can be added. In order for a lexon to be entered, at least one of the terms needs to be articulated first. The terms and roles in lexons can be constrained and the community can then commit to the hybrid ontology by annotating an individual application symbols with a constrained subset of the lexons. At the same time, communities can interact to agree on the equivalence of glosses and the synonymy of terms. Committing to the ontology allows for the data to be explored by other agents via that ontology. Commitments also enable the community to re-interpret the ontology with its extension (i.e. the instances in each annotated system). This will trigger new social processes that lead to a better approximation of the domain, as the community is able to explore the increasingly annotated data, e.g., by formulating queries. Details of a commitment language are found in [31].

3.2 Creation of an Ontology and Annotation of the Data

Based on the possible social interactions provided by and stored in the tool, we developed an ontology⁴ taking in consideration the FOAF⁵, SIOC⁶ and Dublin Core Ontology⁷. This ontology is then used to annotate the relational database of the tool supporting the GOSPL method with D2RQ [4]. D2RQ allows one to map symbols of the relational database to concepts and relations in that ontology as well as providing the data contained in that database as RDF [3] via a SPARQL [26] endpoint.

The ontology we developed captures the social interactions in a hierarchical manner. A first distinction we could make in this ontology is the difference between social processes to start a discussion within one community (e.g., a request to add a lexon) and social processes to start a discussion between communities (e.g. requests to add synonyms). We call such processes respectively intra- and inter-community requests. Both are called *requests* when the distinction does

³ That is, meaningful for that community.

⁴ The ontology can be found on <http://minf.vub.ac.be/ODBASE/ontology.rdf>

⁵ <http://www.foaf-project.org/>

⁶ <http://sioc-project.org/>

⁷ <http://dublincore.org/>

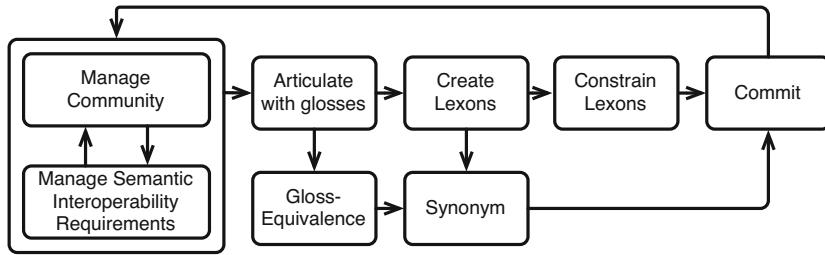


Fig. 1. The GOSPL method [8]

not need to be made. Members of a community can propose changes to the ontology (both formal and informal part) in a forum-like manner and encourages the other stakeholders to express their opinion. The stakeholders can do this by replying (to the proposition or to another reply). Another way for community members to express their opinion is to cast a “vote” in the tool. Thus, next to two types of requests, we also have, *replies* and *votes*. Eventually discussions can also be closed by concluding the discussion. This call is not made by a kind of superuser, but by the community in which the discussion takes place. A member of the community asks to all other users to vote whether or not they agree with the initial proposed request within a certain time frame. That is a reason for also taking the *closing of a topic* into consideration as a social interaction.

During the development of the ontology, we have decided to create subclasses of intra-community requests to relate all requests for a particular part of the ontology (gloss, lexon, constraint,...). This hierarchy will later come in handy for feature selection for the mining algorithm.

We have chosen to develop such an ontology is to allow our framework to reason about the available social processes of different social platforms for user profiling. In other words, the use of ontologies – with notions such as Social Process and Post (from SIOC) – renders our future framework more generative.

3.3 Grouping the Social Interactions

For this paper, we choose to adopt unsupervised learning by applying the K -means mining algorithm. The choice for an unsupervised learning algorithm is motivated by our assumption that user types are a priori not known. A first step in mining is the choice of attributes and data pre-processing. Since the number of features that are chosen should ideally not be too big with respect to the two data sets mentioned in Section 5, we choose to group the social interactions according to the phases in GOSPL. For instance, we grouped all social interactions related to lexons. The result of this grouping leads to a hierarchy of social interactions. A part of this “grouping” of requests is depicted in Fig. 2.

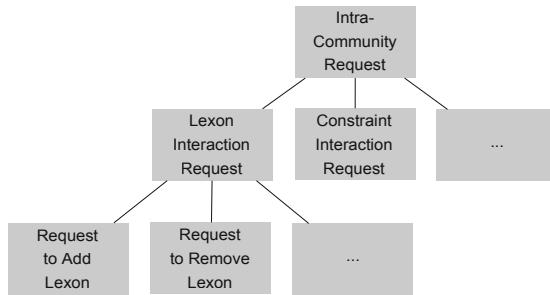


Fig. 2. Graphical depiction of a part of the “grouping” of requests

4 User Profiling the GOSPL Dataset

We apply our ideas on the database of the GOSPL tool for role identification and role composition analysis. The GOSPL tool provides a collection of online forums (called *communities*) in which stakeholders are working together to engineer an ontology. Users post requests to the system on which other users can reply in a natural language what they think about the proposed request. To express their opinion, people can vote indicating whether they (dis)like the proposed request. Based on the outcome of the replies, votes and discussions, one of the stakeholders will close the request by accepting or rejecting it. The result of these interactions will eventually lead to a shared ontology.

We were provided with two datasets of the GOSPL tool. The data stems from two small ontology engineering projects involving respectively 42 and 36 users. These people were asked to work together to develop an ontology about a common concept using the GOSPL tool. Unlike programming fora (see e.g. [28]), where people not necessarily working together can ask questions, we are working here with a more closed setting since we are looking in the field of ontology engineering. Here it is key to find agreements. In the case stakeholders want to propose a new request, they will just start a new thread; that is why e.g. threads are not replied on very often.

We can distinguish three different phases in this user profiling process (see Fig. 3). In the extraction phase we extract the interactions which will be used as the basis for our profiles. In the manipulation phase we apply statistical techniques in order to prepare the data for the final phase where we apply clustering. This last phase is called the clustering phase.

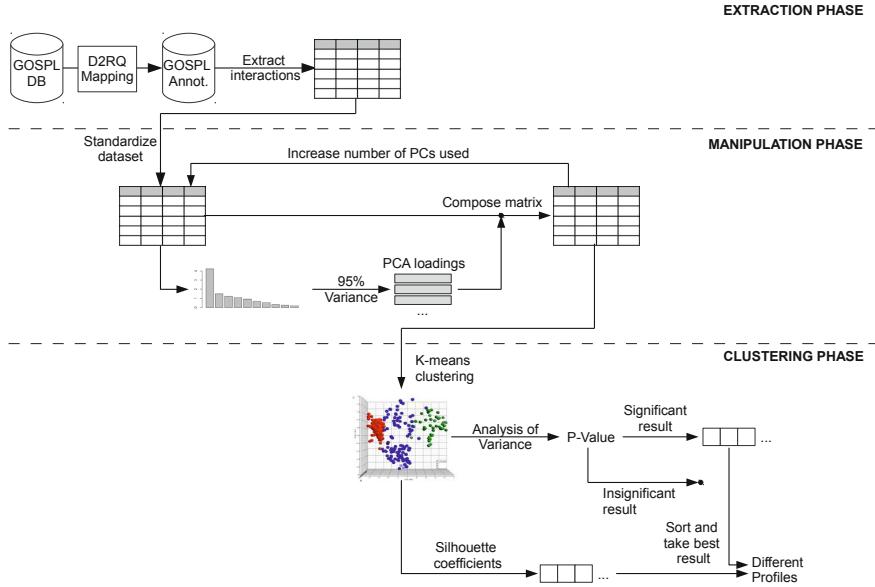


Fig. 3. Graphical depiction of the process to distinguish different types of users

4.1 Extraction Phase

Contrary to what most user profiling applications use, we are only interested in the type of interactions a user makes. That is why in a first phase it is necessary to extract these interactions from the two GOSPL databases. Here we will use the ontology and the D2RQ mapping explained in the previous section, making use of the SPARQL query language.

Important here is that since GOSPL has well over 20 types of social interactions [8], the number of features should ideally not be too big with respect to the number of users. In our case the number of users is respectively 42 and 36. That is why we grouped the social processes according to the phases of the method GOSPL. For instance, we grouped all social interactions related to lexons (see e.g. Fig. 2).

After we extract the interactions per user, we first standardize the data. The reason for standardization is that in a later phase, we will execute a principal component analysis (PCA) [14] that projects data onto directions which maximize the variance. In order to be able to compare variances across different features we standardize the original extracted data.

4.2 Manipulation Phase

In this phase we manipulate the data in order to transform it using a principal component analysis (PCA). In the PCA analysis we look at how many principal

components (PCs) are needed to cover a total cumulative variance of 95% (see e.g [14]).

Once we have found this number N , we compose a new matrix by multiplying the standardized data by the found PCA loadings (see Algorithm 1). We iteratively raise the number of PCs used to transform the dataset ranging from 2 to N . In each iteration we will cluster based upon the newly transformed matrix consisting of the same amount of users, but a reduced amount of dimensions.

```

input : A dataset data
output: A transformed dataset transformedData

1 users  $\leftarrow$  NrOfRows(data);
2 dimensions  $\leftarrow$  NrOfColumns(data);
3 for i  $\leftarrow$  1 to users do
4   for j  $\leftarrow$  1 to dimensions do
5     standardizedData[i, j]  $\leftarrow$   $\frac{\text{data}[i,j] - \mu_j}{\sigma_j}$ 
6   end
7 end
8 pca  $\leftarrow$  ExecutePCA(standardized);
9 loadings  $\leftarrow$  GetRotations(pca);
10 transformedData  $\leftarrow$  (standardizedData  $\times$  loadings);
11 return transformedData;
```

Algorithm 1. Transform data by applying PCA after standardization

4.3 Clustering Phase

In this final phase we use the transformed dataset, after applying standardization and PCA, to cluster the different users based on their interaction behavior. We apply the K -means clustering algorithm on this dataset. Since we do not know the variable K (number of clusters to be found), which is needed for clustering, we iteratively raise K (see Algorithm 2) from 2 to $\lfloor \sqrt{n} \rfloor$, where n represents the number of users. The K -means clustering partitions the data into K mutually exclusive clusters, and returns the index of the cluster to which it has assigned each observation. These cluster results will then be evaluated based on the outcome of an analysis of variance (ANOVA) [1] with a significance level of $\alpha = 0.05$ and an evaluation of the silhouette coefficient belonging to that cluster result.

By performing an ANOVA we test if there is a statistical significant different ($p < \alpha$) between different user groups (profiles) for each of the principal components used to represent the dataset. We will explain this by example: Suppose we work with a reduced dataset of two PCs and $K = 2$. Then we will first perform the ANOVA for the first PC using the groups of the clustering and look if we have a statistical result that is significant. Secondly, we will also do this for the

second PC. If both ANOVA tests are significant, this result in the end will be taken into account after calculating the silhouette coefficients. If one or both tests are insignificant, we will not take this result into account.

We calculate silhouette coefficients to have a measure for internal homogeneity and external heterogeneity between the different clusters. This silhouette coefficient produced is calculated as follows:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \quad (1)$$

Where a_i denotes the average distance to all other items in the same cluster, and b_i is given by calculating the average distance with all other items in each other distinct cluster and then taking the minimum distance. The value of this silhouette coefficient s_i ranges between -1 and 1 , where the former indicates a poor clustering where distinct items are in the same cluster and the latter indicates perfect cluster cohesion and separation. The coefficient thus provides a quality measurement for the cluster method based on how similar intra-cluster items are (cohesion) and how dissimilar inter-cluster items are (separation).

In the end we consider the significant ANOVA tests and look which result has the highest silhouette coefficient. This cluster result is then a basis for distinguishing different profiles that can be used in the continuation of an OE project.

5 User Profiling on the Dataset

As a result of the clustering phase, the algorithm from the previous section provides us with the necessary information to determine how to cluster the user behavior. We present the data here as a matrix where two unknown variables are used, i.e. the number of clusters (variable K) and the number of principal components to be used, respectively represented by rows and columns.

5.1 Dataset I

The first dataset consists of 42 users that produce 9,195 user interactions in total spread over a period of a month. These 42 users worked together to engineer an ontology around the concept *Event* in groups of 4 to 6 users. In a first phase we extract 10 dimensions for these 42 users.

After applying a PCA, we observe that by using 5 principal components of the original 10 we cover a total variance of 95.75%. This means that we can iterate to compose a matrix in the manipulation phase ranging from 2 to 5 dimensions (i.e. principal components). On this composed matrix we apply K -means and observe if the clusters that are found are significant for each of the dimensions. Combined with the silhouette coefficients s_i calculated on the clustering outcome, we obtain Table 1. Finally we sort these results based on the calculated silhouette coefficients only considering the significant results. In this experiment we observe the highest silhouette is obtained, making use of 2 principal components and $K = 5$. This means we distinguish 5 different user profiles in this dataset.

```

input : A transformed dataset data
output: An array sigUserProfiling containing values k and princomp

1 highestSilhouette  $\leftarrow$  0;
2 users  $\leftarrow$  NrOfRows(data);
3 maxPCs  $\leftarrow$  InterestingPCs(data) ; /* TotalVar(PCs)  $\geq$  0.95 */
4 maxK  $\leftarrow$   $\lceil \text{Sqrt}(\text{users}) \rceil$ ;
5 for i  $\leftarrow$  1 to maxPCs do
6   reduced  $\leftarrow$  TakeFirstColumns(data,i);
7   for j  $\leftarrow$  2 to maxK do
8     clusters  $\leftarrow$  PerformKMeans(reduced,i);
9     aov  $\leftarrow$  PerformANOVA(clusters);
10    significance  $\leftarrow$  PValue(aov);
11    if significance  $\leq$  0.05 then
12      silhouette  $\leftarrow$  CalculateSilhouette(clusters);
13      if silhouette  $>$  highestSilhouette then
14        highestSilhouette  $\leftarrow$  silhouette;
15        princomp  $\leftarrow$  i;
16        k  $\leftarrow$  j;
17      end
18    end
19  end
20 end
21 sigUserProfiling  $\leftarrow$  Array(k,princomp);
22 return sigUserProfiling;

```

Algorithm 2. Look for most significant user profiling**Table 1.** Output of clustering phase for experiment 1

	2 PCs		3 PCs		4 PCs		5 PCs	
	Sig?	<i>s_i</i>	Sig?	<i>s_i</i>	Sig?	<i>s_i</i>	Sig?	<i>s_i</i>
<i>K</i> = 2	N	0.8465052	N	0.8211668	N	0.7990878	N	0.7919471
<i>K</i> = 3	Y	0.4808486	N	0.394155	N	0.3447474	N	0.3289679
<i>K</i> = 4	Y	0.5063455	Y	0.4376691	Y	0.4064185	N	0.3833227
<i>K</i> = 5	Y	0.5151144	Y	0.4467786	N	0.4584918	N	0.4640916
<i>K</i> = 6	Y	0.4869484	Y	0.4174907	Y	0.4909751	N	0.3957277

The columns *Sig?* represent if the result is significant (Y) or not (N).
The columns *s_i* represent the silhouette coefficient belonging to the cluster result.

5.2 Dataset II

In a second dataset 36 users of the GOSPL tool cooperated to constitute a common ontology considering the concept *Scientific publications*. In total these users worked together for two month, resulting in 7,127 interactions in total. Like the first dataset, these users worked together in groups of 4 to 6 users. In a first phase we extract 11 dimensions⁸ for 36 users. In the manipulation phase, we observed that by using the first 8 principal components we cover 95.37% of the total variance in the dataset. Now we can use these to compose a new matrix in the manipulation phase ranging from 2 to 8 dimensions (i.e. principal components). K -means clustering and silhouette coefficient calculations result in Table 2. After sorting out the *best* result, based on the significance and the silhouette coefficients, we observed that using 2 PCs and parameter $K = 3$ results in the highest silhouette coefficient.

Table 2. Output of clustering phase for experiment 2

	2 PCs		3 PCs		4 PCs		5 PCs	
	Sig?	s_i	Sig?	s_i	Sig?	s_i	Sig?	s_i
$K = 2$	N	0.6479863	N	0.5808876	N	0.5964199	N	0.5724287
$K = 3$	Y	0.5469009	N	0.4733644	N	0.5004817	N	0.5122605
$K = 4$	Y	0.546263	N	0.41647	N	0.4487667	N	0.4988729
$K = 5$	Y	0.5389498	Y	0.495092	Y	0.4508551	N	0.4310294
$K = 6$	Y	0.5284029	Y	0.4758109	Y	0.4312709	Y	0.4264906

	6 PCs		7 PCs		8 PCs	
	Sig?	s_i	Sig?	s_i	Sig?	s_i
$K = 2$	N	0.5518503	N	0.5387554	N	0.5273153
$K = 3$	N	0.4508124	N	0.3132408	N	0.399256
$K = 4$	N	0.4734011	N	0.3700723	N	0.4452684
$K = 5$	N	0.4449335	N	0.3821288	N	0.3940165
$K = 6$	N	0.3909784	N	0.4202661	N	0.4101321

The columns *Sig?* represent if the result is significant (Y) or not (N).
The columns s_i represent the silhouette coefficient belonging to the cluster result.

5.3 Interpretation of User Profiles

We presented a method for finding different user profiles, by clustering based on the user interactions. However, what is the meaning of these different types of users? Since we cannot identify the characteristics of these clusters automatically, we should interpret them. This can be done observing differences between the averages of the original data for the different user profiles.

⁸ Note that for the same structured dataset we can now extract 11 dimensions since here the semantic interoperability required is now captured in the GOSPL tool, whereas in the first dataset these requirements were not explicitly captured.

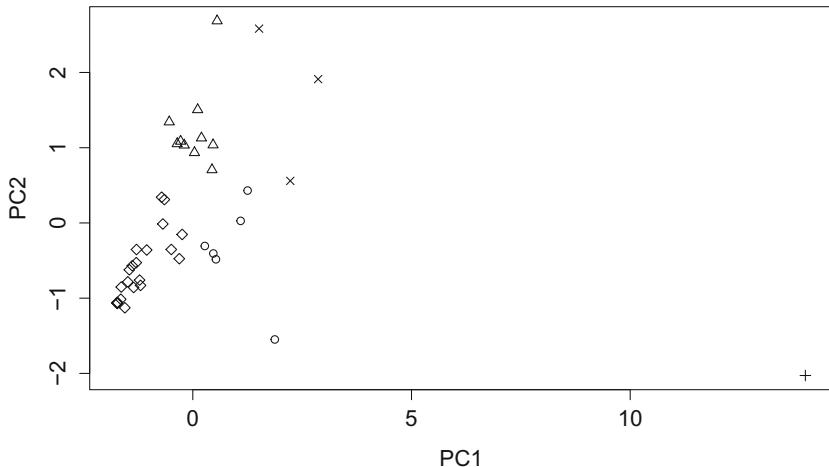


Fig. 4. Clusters of dataset I (clusters indicated with symbols)

User Profiles in Dataset I. In this dataset we observe 5 distinct profiles in Fig. 4. We interpret them by looking at descriptive statistics of each cluster (see Table 3) and observe the following differences between these user types:

- We interpret *cluster 1* (○ symbol) and *cluster 4* (× symbol) together since both clusters have similar behavior for nearly all of the dimensions. On overall, we consider these two groups as active users of the GOSPL tool. However, we do observe that cluster 4 is more familiar with expressing his personal opinion than cluster 1 by replying on a certain topic or by casting votes, respectively dimensions 8 and 10.
- *Cluster 2* (△ symbol) is less active than the previous mentioned user profiles. However, we do observe this user profile prefers interacting with GOSPL by replying, concluding discussions and casting votes, respectively dimensions 8, 9 and 10.
- *Cluster 3* (+ symbol) exists of only one user in this dataset. This user instance has the maximum amount of user interactions for nearly each dimension. In the plots we observe, this instance behaves extremely different than the others.
- *Cluster 5* (◊ symbol) can be considered as the more passive group of users, since this user profile has the lowest amount of user interactions.

User Profiles in Dataset II. In dataset II we observed 3 different types of users in Fig. 5. We interpret them by looking at descriptive statistics of each cluster (see Table 4) and observe the following differences between these two user types:

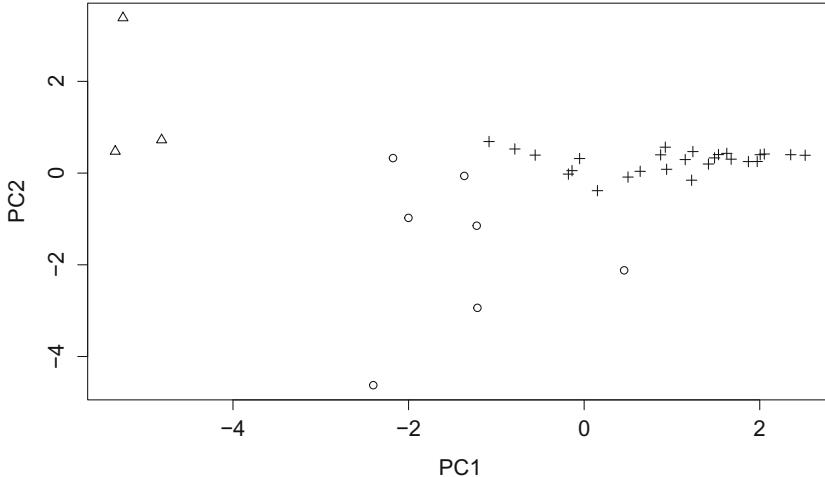


Fig. 5. Clusters of dataset II (clusters indicated with symbols)

- *Cluster 1* (\triangle symbol) has a higher amount of interactions as the other two user profiles for more than half of the dimensions. We consider this cluster as being the active users.
- *Cluster 2* (\circ symbol) and *cluster 3* ($+$ symbol) both have a lower interaction amount than cluster 1. However, we consider them as separate groups of users since cluster 2 has in general⁹ more interactions than cluster 3. Note that cluster 2 is very active in making interactions considering lexons.

6 Discussion and Conclusion

In literature the identification process of different profiles often makes use of both quantitative and qualitative features [28]. On the contrary, in this paper we demonstrated a method for identifying user profiles using the interactions of the users as a single input on two datasets. We started by using a semantic mapping on top of the GOSPL database. For this mapping, we made an extension of the SIOC Core ontology. We use this ontology, since the concept of posts on an online platform is widely used, moreover the ontology is easy to reuse. The benefit of using a semantic mapping is that when we apply this mapping to multiple datasets (even from various platforms), we use a uniform way for identifying the different interactions.

After we obtained the quantitative number of interactions by each user, we pre-process the datasets. Since we deal with unequal means and variances we decided to standardize each dimension. Then we applied a PCA analysis in order

⁹ Dimensions 4, 5, 6 and 7 have very low interaction amounts over the complete dataset.

to obtain the most interesting principal components for reducing the dimensionality of the data. In our datasets we originally used datasets of 10 and 11 dimensions. After reduction of the dimensionality, we ended up with a two-dimensional dataset in both cases.

The method demonstrated the first dataset has five user profiles, while the second dataset only has three distinct user profiles. Though the second dataset covered a longer time frame in which the users of GOSPL could interact, we discovered less distinct user profiles using the K -means clustering technique. In the second dataset, we observed the differences were smaller between the found user profiles. This can partly be explained by the lower number of interactions made by users in the second dataset. Moreover the three active users (found in cluster 1) in this second dataset were responsible for 1,533 interactions which is over 20% of the total interaction amount.

Since we only look at the quantitative measurements of interactions as a basis, this method is very sensitive to distinguishing active users as a separate user profile. By labelling users as active and passive, we can raise doubts about the effectiveness of the passive users for the ontology engineering project. However, we believe that the passive users –read less active users– deliver input that is equally valuable as the ones of the active users. Each interaction triggers an other interaction, e.g. an acception of a lexon, will trigger an interaction to constrain that lexon using the GOSPL method.

In this proposed method we validate the cluster quality using silhouette coefficients. We choose this cluster validity technique since this technique provides a good measurement that combines both internal homogeneity and external heterogeneity of clusters. Other validity techniques that could have been used are *Dunn index*, *Davies-Bouldin index*, and the *C-index* [12]. The silhouette coefficient is bounded between -1 for incorrect clustering and $+1$ for highly dense clustering. We are thus interested in clustering with this coefficient as high as possible.

We consider this paper as a first step in assigning roles automatically to the users based on the behavior of the user. The project leader of an ontology engineering project decides when to run this method, and after analysis of the resulting profiles he can rearrange his team of ontology engineers in order to work more efficiently. This can either be by composing task groups of a similar user type or by composing new task groups by combining different user types.

Acknowledgements. We like to thank Prof. Dr. Meersman R. and Dr. Debruyn C. for the provided datasets and input on GOSPL, and the statistical team in our department (Prof. Dr. Buyl R., Prof. Dr. Coomans D., Prof. Dr. Questier F. and Simons K.).

References

1. Armstrong, R., Slade, S., Eperjesi, F.: An introduction to analysis of variance (anova) with special reference to data from clinical experiments in optometry. *Ophthalmic and Physiological Optics* 20(3), 235–241 (2000)

2. Billsus, D., Pazzani, M.J.: A personal news agent that talks, learns and explains. In: Proceedings of the Third Annual Conference on Autonomous Agents, AGENTS 1999, pp. 268–275. ACM, New York (1999)
3. Brickley, D., Guha, R.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, W3C (2004)
4. Cyganiak, R., Bizer, C., Garbers, J., Maresch, O., Becker, C.: The D2RQ mapping language (2012), <http://d2rq.org/d2rq-language>
5. De Leenheer, P., Christiaens, S., Meersman, R.: Business semantics management: A case study for competency-centric HRM. Computers in Industry 61(8), 760–775 (2010)
6. De Leenheer, P., Debruyne, C., Peeters, J.: Towards social performance indicators for community-based ontology evolution. In: Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK 2008), collocated with the 8th International Semantic Web Conference (ISWC 2009), CEUR-WS (2009)
7. Debruyne, C., Meersman, R.: Semantic interoperation of information systems by evolving ontologies through formalized social processes. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 444–459. Springer, Heidelberg (2011)
8. Debruyne, C., Meersman, R.: GOSPL: A method and tool for fact-oriented hybrid ontology engineering. In: Morzy, T., Härder, T., Wrembel, R. (eds.) ADBIS 2012. LNCS, vol. 7503, pp. 153–166. Springer, Heidelberg (2012)
9. Falconer, S.M., Tudorache, T., Noy, N.F.: An analysis of collaborative patterns in large-scale ontology development projects. In: Musen, M.A., Corcho, Ó. (eds.) K-CAP, pp. 25–32. ACM (2011)
10. Golder, S.A., Donath, J.: Social roles in electronic communities. In: Proc. of Association of Internet Researchers Conference (2004)
11. Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies 43, 907–928 (1993)
12. Günter, S., Bunke, H.: Validation indices for graph clustering. Pattern Recogn. Lett. 24(8), 1107–1113 (2003)
13. Hautz, J., Hutter, K., Fuller, J., Matzler, K., Rieger, M.: How to establish an online innovation community? the role of users and their innovative content. In: Proceedings of the 2010 43rd Hawaii International Conference on System Sciences, HICSS 2010, pp. 1–11. IEEE Computer Society, Washington, DC (2010)
14. Jolliffe, I.: Principal Component Analysis. Springer (1986)
15. Khatri, V., Brown, C.V.: Designing data governance. Commun. ACM 53(1), 148–152 (2010)
16. Kotis, K., Vouros, G.A.: Human-centered ontology engineering: The HCOME methodology. Knowledge Information Systems 10(1), 109–131 (2006)
17. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the 12th International Machine Learning Conference, ML 1995 (1995)
18. Lieberman, H.: Letizia: an agent that assists web browsing. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 1995, vol. 1, pp. 924–929. Morgan Kaufmann Publishers Inc., San Francisco (1995)
19. Lieberman, H., Van Dyke, N.W., Vivacqua, A.S.: Let's browse: a collaborative web browsing agent. In: Proceedings of the 4th International Conference on Intelligent User Interfaces, IUI 1999, pp. 65–68. ACM, New York (1999)
20. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering 7(1), 76–80 (2003)

21. McDonald, D.W., Ackerman, M.S.: Expertise recommender: a flexible recommendation system and architecture. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, CSCW 2000, pp. 231–240. ACM, New York (2000)
22. Middleton, S.E., Shadbolt, N.R., De Roure, D.C.: Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.* 22(1), 54–88 (2004)
23. Nolker, R.D., Zhou, L.: Social computing and weighting to identify member roles in online communities. In: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2005, pp. 87–93. IEEE Computer Society, Washington, DC (2005)
24. Pinto, H.S., Staab, S., Tempich, C.: DILIGENT: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In: de Mántaras, R.L., Saitta, L. (eds.) ECAI, pp. 393–397. IOS Press (2004)
25. Pontikakos, C., Zakynthinos, G., Tsiligiridis, T.: Designing cscw system for integrated, web-based, cotton cultivation services. *Operational Research* 5(1), 177–191 (2005)
26. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation, W3C (2008)
27. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, CSCW 1994, pp. 175–186. ACM, New York (1994)
28. Rowe, M., Fernández, M., Angeletou, S., Alani, H.: Community analysis through semantic rules and role composition derivation. *J. Web Sem.* 18(1), 31–47 (2013)
29. Schafer, J.B., Konstan, J.A., Riedl, J.: E-commerce recommendation applications. *Data Min. Knowl. Discov.* 5(1-2), 115–153 (2001)
30. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating word of mouth. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 1995, pp. 210–217. ACM Press/Addison-Wesley Publishing Co., New York (1995)
31. Verheyden, P., De Bo, J., Meersman, R.: Semantically unlocking database content through ontology-based mediation. In: Bussler, C.J., Tannen, V., Fundulaki, I. (eds.) SWDB 2004. LNCS, vol. 3372, pp. 109–126. Springer, Heidelberg (2005)

A Descriptive Statistics of Datasets

Table 3. Descriptive statistics of clusters in first dataset

	dim 1	dim 2	dim 3	dim 4	dim 5	dim 6	dim 7	dim 8	dim 9	dim 10
cluster 1 (n = 6)	\bar{x}	25.67	28.50	16.67	9.50	0.33	9.33	0.67	13.50	69.33
	σ	10.03	12.23	9.67	9.71	0.82	6.02	1.63	12.55	19.63
	95% CI lower bound	15.14	15.67	6.52	-0.69	-0.52	3.01	-1.05	0.33	48.73
	95% CI upper bound	36.20	41.33	26.81	19.69	1.19	15.65	2.38	26.67	89.94
cluster 2 (n = 10)	\bar{x}	14.00	11.40	9.30	0.20	0.60	1.40	1.20	35.60	34.70
	σ	8.30	3.68	14.13	0.63	0.97	2.12	1.23	14.86	17.52
	95% CI lower bound	8.06	8.76	-0.81	-0.25	-0.09	-0.12	0.32	24.97	22.17
	95% CI upper bound	19.94	14.04	19.41	0.65	1.29	2.92	2.08	46.23	47.23
cluster 3 (n = 1)	\bar{x}	179.00	112.00	123.00	49.00	2.00	82.00	3.00	97.00	570.00
	σ	-	-	-	-	-	-	-	-	-
	95% CI lower bound	-	-	-	-	-	-	-	-	-
	95% CI upper bound	-	-	-	-	-	-	-	-	-
cluster 4 (n = 3)	\bar{x}	32.00	36.33	29.00	6.67	0	10.67	2.00	50.00	108.00
	σ	11.79	15.95	19.29	6.11	0	3.06	2.65	23.07	54.62
	95% CI lower bound	2.71	-3.28	-18.91	-8.51	0	3.08	-4.57	-7.30	-27.68
	95% CI upper bound	61.29	75.95	76.91	21.85	0	18.26	8.57	107.30	243.68
cluster 5 (n = 22)	\bar{x}	7.27	5.73	2.14	0.36	0.09	0.82	0.09	7.18	10.59
	σ	9.16	6.42	3.98	0.95	0.43	1.59	0.29	9.68	12.56
	95% CI lower bound	3.21	2.88	0.37	-0.06	-0.10	0.11	-0.04	2.89	5.02
	95% CI upper bound	11.33	8.57	3.90	0.79	0.28	1.52	0.22	11.48	16.16

Table 4. Descriptive statistics of clusters in second dataset

	dim 1	dim 2	dim 3	dim 4	dim 5	dim 6	dim 7	dim 8	dim 9	dim 10	dim 11
cluster 1 (n = 3)	\bar{x}	26.33	19.67	21.33	0	0	0	1.33	82.33	95.33	250.33
	σ	8.08	9.61	8.33	0	0	0	2.31	7.09	18.50	133.99
	95% CI lower bound	6.25	-4.20	0.65	0	0	0	-4.40	64.71	49.37	-82.53
	95% CI upper bound	46.41	43.54	42.02	0	0	0	7.07	99.96	141.30	583.19
cluster 2 (n = 7)	\bar{x}	6.57	20.71	12.00	0.43	0.57	5.71	0	41.71	38.57	132.29
	σ	5.97	16.49	11.25	0.79	0.98	6.68	0	22.93	16.21	64.19
	95% CI lower bound	1.05	5.46	1.59	-0.30	-0.33	-0.46	0	20.51	23.58	72.92
	95% CI upper bound	12.09	35.97	22.41	1.16	1.47	11.89	0	62.92	53.56	191.65
cluster 3 (n = 26)	\bar{x}	3.08	6.85	3.62	0.42	0	0.23	0.04	14.31	16.35	94.65
	σ	4.07	6.87	5.73	1.39	0	0.65	0.20	13.40	19.21	56.31
	95% CI lower bound	1.43	4.07	1.30	-0.14	0	-0.03	-0.04	8.90	8.59	71.91
	95% CI upper bound	4.72	9.62	5.93	0.99	0	0.49	0.12	19.72	24.11	117.40

- dim 1: interactions about glosses
- dim 2: interactions about lexons
- dim 3: interactions about constraints
- dim 4: interactions about supertype relations
- dim 5: interactions about equivalence between glosses
- dim 6: interactions about synonyms
- dim 7: interactions considering general requests
- dim 8: interactions considering replies
- dim 9: interactions to close topics
- dim 10: interactions about casting votes
- dim 11: interactions about semantic interoperability requirements

Defining and Investigating the Scope of Users and Hashtags in Twitter^{*} (Short Paper)

Daniel Leggio, Giuseppe Marra, and Domenico Ursino

DIIES, University Mediterranea of Reggio Calabria, Via Graziella,
Località Feo di Vito, 89122 Reggio Calabria, Italy

Abstract. In this paper we aim at analyzing the scope of an entity in Twitter. In particular, we want to define a framework for measuring this scope from multiple viewpoints (e.g., influence, reliability, popularity) simultaneously and for multiple entities (e.g., users, hashtags). In this way, we can compare different properties and/or different entities. This comparison allows the extraction of knowledge patterns (for instance, the presence of anomalies and outliers) that can be exploited in several application domains (for example, information diffusion).

1 Introduction

The scope of an entity within a social network represents the extension of the area in which it is relevant. Scope is an extremely important factor when one wants to analyze how much an entity conditions the social environment around it. Scope investigation can be performed w.r.t. several social properties, for instance social influence (i.e., how much an entity stimulates its neighbors to behave in a similar way) or reliability (i.e., how much the content provided by an entity is valuable for its neighbors). The usefulness of this information is extremely valuable in very different fields of social research, for instance in the analysis of how information spreads within a social network, or in determining the trust and the reputation of a user in a network [5].

In the past, a lot of effort has been put in the computation of specific features expressing the conditioning of a social network entity (e.g., a user) on its neighbors. For instance, as for the evaluation of user influence, [4] proposes to exploit the well known *PageRank* algorithm to assess the distribution of influence throughout the network. With the aim of taking not only network topology but also user behavior into account, [12] develops Influence Passivity, a graph-based approach, similar to the HITS algorithm, which assigns a relative influence and a passivity score to each user, based on the fraction of information forwarded by her. Following a similar idea, [16] proposes TwitterRank, an evolution of PageRank that measures user influence by considering both the topics of interest for users and link structure. A totally different approach is adopted by [7] that characterizes influential users by tackling the influence maximization problem [6].

* This work was partially supported by Aubay Italia S.p.A.

If we move from user influence to user reliability, [14] introduces a spam detection prototype system to identify suspicious users on Twitter. Here, the authors define a directed social graph model to explore the follower-friend relationships among users by means of Bayesian classification. Following the same research line, [11] faces the spam detection problem via both classification and clustering, whereas [2] exploits Support Vector Machine.

Past research efforts have been put to investigate not only users but also other social entities, such as hashtags and tweets. For instance, as for the evaluation of hashtag popularity in Twitter, [17] studies temporal patterns associated with hashtags and investigates how hashtag popularity varies over time. Similar researches were conducted by [8] and [9]. Facing the same problem from another viewpoint, [15] automatically computes the overall sentiment popularity for a given hashtag in a certain time period. A very common analysis in Twitter regards the evaluation of tweet and hashtag reliability, especially during emergency situations, when it is essential to find correct information. To this end, [10] explores the behavior of Twitter users during the Chile earthquake with the aim of distinguishing false rumors from confirmed news and of analyzing how they spread within the network. Similar analyses can be found in [1] for the Japan tsunami, where retweet rate was exploited as trust indicator, and in [13] for the Fukushima disaster, where information source is the element distinguishing rumors from important news.

In this paper we aim at providing a contribution in this setting. In fact, even in this case, we want to analyze the scope of an entity in a network. However, we want to investigate this problem at a higher abstraction level, in such a way as to define a framework to measure this scope from *more viewpoints simultaneously* and for *more entity types*. In this way we aim at overcoming the limitations of the previous proposals. Indeed, the definition of specific approaches to evaluating a single property often leads to systems whose extension to other properties is very hard or totally impossible. Furthermore, a more complete approach should simultaneously consider more properties allowing the investigation of the same phenomenon under different viewpoints and, consequently, the extraction of more complete and powerful knowledge. Finally, taking more entity types into account provides a more complete, powerful and realistic picture of the relationships among the different entities of a social network and of the scope of each of them w.r.t. the others in the network. Currently, we specialized our approach to Twitter, even if, in the future, we plan to further increase its abstraction level in such a way as to let it operate on a wide range of social networks.

In our research we operated as follows. First, we analyzed the API of Twitter to consider all the information that can be extracted from this network. Then, we proceeded to define the concept of scope of an entity in Twitter. For this purpose we preliminarily defined a support graph-based model for Twitter, a support metric and some support functions allowing the scope of an entity to be analyzed under different viewpoints. Then, we proceeded with scope computation. After having chosen two properties to investigate (i.e., influence and

Table 1. Subset of nodes and edges in a Twitter multi-graph

Name	Symbol	Semantics
User nodes	V_u	$u \in V_u$ represents a user
Tweet nodes	V_t	$t \in V_t$ represents a tweet
Hashtag nodes	V_h	$h \in V_h$ represents a hashtag
Following edges	E_{uu}^{fw}	$(u_i, u_j) \in E_{uu}^{fw}$ indicates that the user u_i is following the user u_j
List membership edges	E_{uu}^l	$(u_i, u_j) \in E_{uu}^l$ indicates that u_i is member of a list of u_j
Authorship edges	E_{ut}^a	$(u, t) \in E_{ut}^a$ indicates that the user u is the author of the tweet t
Retweet edges	E_{ut}^r	$(u, t) \in E_{ut}^r$ indicates that u retweeted t
Favorite edges	E_{ut}^{fv}	$(u, t) \in E_{ut}^{fv}$ indicates that u specified a preference for t
Mention edges	E_{ut}^m	$(u, t) \in E_{ut}^m$ indicates that u is mentioned in t
Reply edges	E_{tt}^r	$(t_i, t_j) \in E_{tt}^r$ indicates that the tweet t_i is a reply to the tweet t_j
Hashtag edges	E_{th}	$(t, h) \in E_{th}$ indicates that the tweet t contains the hashtag h

reliability), we defined our scope investigation framework and specialized it to these two properties.

Finally, we performed an experimental campaign aiming at verifying the suitability of the proposed approach. This campaign allowed us to extract a lot of knowledge about influence and reliability of users and hashtags in Twitter. This knowledge is presented in detail in the last part of this paper.

2 Scope Definition

In this section, we define the concept of scope of an entity in Twitter. For this purpose we need: (i) a support model, capable of representing and handling all necessary information; (ii) some support functions, representing different viewpoints from which the scope can be seen; (iii) a support metric, allowing the intensity of the scope to be measured. Our support model, called *Twitter multi-graph*, stores all information needed to define and investigate the scope of entities in Twitter. This information can be derived from the API of Twitter. A Twitter multi-graph $G_T = \langle V_T, E_T \rangle$ consists of a set V_T of nodes and a set E_T of directed edges of the form (v_i, v_j) , where v_i is the source node and v_j is the target one. There are 3 subsets of nodes and 8 subsets of edges. Their name, symbol and semantics are reported in Table 1.

Support functions allow the investigations of the scope of a user or a hashtag in Twitter from different viewpoints. In the following we illustrate in detail two of them, e.g. reliability and influence. However, we point out that, whenever necessary, new functions can be defined in a similar fashion. For each function, first we specify its general meaning. Then, we denote the different node types it can be applied to. Finally, for each node pair, we specify the set of parameters the corresponding computation depends on. All these parameters can be obtained from the API of Twitter and, consequently, from the Twitter multi-graph.

The two support functions we illustrate in detail are:

- $reliability(\rho_{v_i, v_j})$; it expresses how much v_i relies on v_j . In particular, (v_i, v_j) could be a pair of users (u_i, u_j) or a user-hashtag pair (u, h) .

- ρ_{u_i, u_j} depends on: (i) the number of times u_i inserted the tweets of u_j in her favorites; (ii) the presence of a following relationship from u_i to u_j ; (iii) the presence of u_j in the lists of u_i .
- $\rho_{u, h}$ indicates the trustworthiness of h for u ; it depends on: (i) the number of the tweets of u containing h ; (ii) the number of times u added a tweet with h to her favorites.
- $influence(\iota_{v_i, v_j})$; it measures how much the activities of v_i inspire those of v_j . In particular, (v_i, v_j) could be a pair of users (u_i, u_j) , a pair hashtag-user (h, u) or a pair of hashtags (h_i, h_j) .
 - ι_{u_i, u_j} expresses how much the behavior of u_i influences that of u_j . It depends on: (i) the number of times u_j retweeted, replied to and/or added to her favorites the tweets of u_i ; (ii) the presence of u_i in a list of u_j ; (iii) the existence of a following relationship from u_j to u_i ; (iv) the number of times u_i is mentioned in the tweets of u_j .
 - $\iota_{h, u}$ denotes how much h inspires the behavior of u . It depends on: (i) the number of times u retweeted, replied to and/or added to her favorites a tweet containing h ; (ii) the number of times u wrote tweets containing h .
 - ι_{h_i, h_j} indicates how much the reception of h_i inspires a user to exploit h_j . It depends on: (i) the number of times a user sent a reply containing h_j to a tweet containing h_i ; (ii) the number of times h_i and h_j appeared together in the same tweets.

Our support metric has been conceived to measure the intensity of the scope of a user or a hashtag in Twitter. Our purpose has been to define a very general metric that can be applied to any support function. However, we have seen that support functions can be very heterogeneous. As a consequence, it is necessary to conciliate function heterogeneity with the advisability to have a unique support metric. Thus, we need to construct a unique support structure (we call *Metric Graph*, or m-graph, this structure) independent of the specific support function into consideration. To this end, each support function needs to be coupled with a set of rules capable of constructing a Metric Graph whose semantics is still related to this function but whose analysis can be performed in a uniform way for all functions.

To better face the complexity of this task, it is advisable to adopt a two-step process. In the first step a new multi-graph, called *Compressed Twitter multi-graph* G_C , is constructed. It allows a much quicker evaluation of support functions. In the second step a Metric Graph G_M is constructed for each support function of interest.

Given a Twitter multi-graph $G_T = \langle V_T, E_T \rangle$, the corresponding Compressed Twitter multi-graph $G_C = \langle V_C, E_C \rangle$ consists of a set V_C of nodes and a set E_C of directed edges of the form (v_i, v_j, w_{ij}) , where v_i is the source node, v_j is the target node and w_{ij} is a positive integer. Intuitively, a Compressed Twitter multi-graph G_C can be obtained from a Twitter multi-graph G_T by removing tweet nodes and by suitably aggregating the edges of the same type.

Given a Twitter multi-graph $G_T = \langle V_T, E_T \rangle$, the corresponding m-graph $G_M = \langle V_M, E_M \rangle$ consists of a set V_M of nodes and a set E_M of directed edges. $V_M = V_u \cup V_h$. V_u (resp., V_h) is the set of user (resp., hashtag) nodes; it is a subset of the user (resp., hashtag) nodes of G_T , possibly coincident with it. An element $(v_i, v_j, w_{ij}) \in E_M$ represents an edge from v_i to v_j ; w_{ij} is a weight, in the real interval $[0, 1]$, representing the “impact” that v_i has on v_j .

Observe that G_M is a unique data structure that can be exploited for each support function. As a consequence, given a Twitter multi-graph G_T , there is an m-graph G_M^F for each support function F into consideration. The specificity of F is taken into account in the computation of the weights of the edges of G_M^F . Indeed, the values of these weights depend on G_T and F .

Our metric aims at allowing the measurement of the “distance” [3] of two nodes v_i and v_j in an m-graph, say G_M^F , associated with a support function F . Intuitively, given two pairs of nodes (v_i, v_j) and (v_i, v_k) , if the distance between v_i and v_j is lower than that between v_i and v_k , then the scope of v_i on v_j should be higher than that of v_i on v_k . Thus, we need a metric relating a short distance to a high value of scope. Taking into account that the weights in G_M^F are in the real interval $[0, 1]$, it is easy to note that the higher the product of the weights of the edges of the path from v_i to v_j , the lower the distance of v_j from v_i .

This intuition is formalized as follows: let $G_M^F = \langle V_M, E_M \rangle$ be an m-graph associated with a support function F and let v_i and v_j be two nodes in G_M^F connected by q paths $\{p_{ij}^1, \dots, p_{ij}^q\}$. Finally, let p_{ij} be a path from v_i to v_j . (i) The *impact* $\Psi(p_{ij})$ of v_i on v_j constrained by the path p_{ij} is defined as the product of the weights of the edges composing p_{ij} . (ii) The *impact* $\Psi(v_i, v_j)$ of v_i on v_j is defined as: $\Psi(v_i, v_j) = \max_{h=1..q} \Psi(p_{ij}^h)$ (iii) The *distance* $d(v_i, v_j)$ from v_i to v_j is defined as the length of the path p_{ij} associated with the impact $\Psi(v_i, v_j)$ of v_i on v_j .

Let $G_M^F = \langle V_M, E_M \rangle$ be an m-graph associated with a support function F and let v be a node of G_M^F . The k^{th} neighborhood $nbh(v, k)$ of v in G_M^F is defined as $nbh(v, k) = \{v' \mid v' \in V_M, v' \neq v, d(v, v') = k\}$

Given a Twitter multi-graph G_T , a support function F and the corresponding m-graph $G_M^F = \langle V_M, E_M \rangle$, the computation of the scope of a node $v \in V_M$ is performed as follows. First, all the possible (non-empty) neighborhoods $nbh(v, k)$ of v are defined. Then, $F(v, k)$, i.e., the scope of v in its k^{th} neighborhood w.r.t. F , is computed for each identified neighborhood. In particular, for each non-empty neighborhood $nbh(v, k)$, $F(v, k)$ is computed as $F(v, k) = \sum_{v' \in nbh(v, k)} \Psi(v, v')$.

3 Scope Investigation

The theoretical framework we have defined in Section 2 is characterized by a high abstraction level, thanks to the presence of support functions. In fact, these allow the scope of a user or a hashtag to be seen under different viewpoints, in a multi-dimensional logic, but in a uniform fashion. The high abstraction level characterizing our framework does not affect its capability of operating with a high level of detail; in fact, this last feature is guaranteed by the fact that our

framework handles all information that can be extracted through the API of Twitter. Thanks to the features mentioned above, the scope investigation tasks that can be designed and implemented are almost unlimited. Each investigation, in turn, allows the extraction of one or more knowledge patterns. To give an idea of the possible investigations that can be performed through our platform, in this section we describe one of them. It regards the variation of the scope of a node (corresponding to a user or a hashtag) when moving away from it in the Twitter graph. This corresponds to study $F(v, k)$ against k . More formally, let $G_M^F \langle V_M, E_M \rangle$ be an m-graph associated with a Twitter multi-graph G_T . For each node $v \in V_M$, we compute $F(v, k)$, for each k ranging from 0 to a certain value \bar{k} , indicating the maximum value of interest for this investigation. After this, we compute $\bar{F}(k) = \frac{\sum_{v \in V_M} F(v, k)}{|V_M|}$.

Observe that, if in a chart we represent k in the x-axis and $\bar{F}(k)$ in the y-axis, we obtain a broken line. There could exist several general template patterns that could be identified for this line. Some very common ones are: constant, irregular, logarithmic, linear, polynomial, exponential. Furthermore, all of them, except the first one, can be increasing or decreasing. Each of these patterns provides very important information about the scope, and ultimately the role and the impact, of the corresponding node in Twitter.

To test this idea we performed an experiment on a fragment of Twitter. In particular, our Twitter multi-graph G_T consisted of 1,218,911 nodes and 2,580,845 edges. The dataset can be downloaded from the address <http://www.barbiana20.unirc.it/Scope0dbase2014.html>. The password the Reader must specify is "85749236". We conducted our experimental campaign from March 20th, 2014 to June 8th, 2014.

We computed the variation of $\bar{F}(k)$ against k for the two support functions introduced in Section 2, i.e., *influence* and *reliability*. Obtained results are reported in Figure 1.

From the analysis of this figure it clearly emerges that both the influence and the reliability of a node (and, consequently, of the corresponding user or hashtag) decrease with a power law when k increases. Now, it was easily intuitable that the

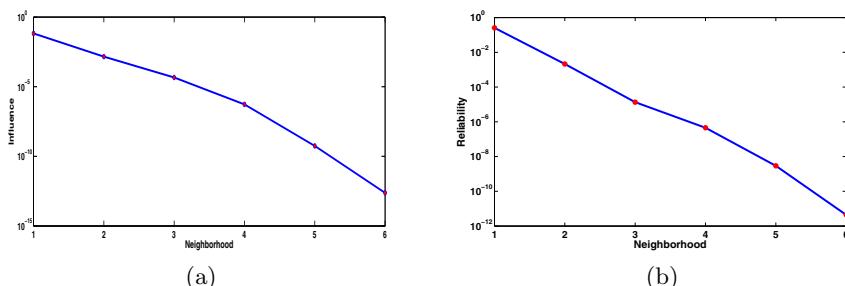


Fig. 1. (a) Variation of the influence $\iota(k)$ against the neighborhood degree k - logarithmic scale; (b) Variation of the reliability $\rho(k)$ against the neighborhood degree k - logarithmic scale

influence and/or the perceived reliability of a user or a hashtag decreases when going away from it in a Twitter graph. Obtained results confirm this intuition but supply much more information since they indicate that this decrease is dramatic because it follows a power law. As a practical consequence, the “weight” of a user in Twitter is substantially limited to its direct neighbors and to the direct neighbors of these last ones. This is an example of a knowledge pattern that can be derived thanks to our approach. It is a precious knowledge that has an enormous importance when organizing an information diffusion campaign or in other similar circumstances.

4 Conclusion

In this paper we have proposed an approach aiming at defining and investigating the scope of users and hashtags in Twitter. First we have proposed a graph-based model, some support functions and a support metric allowing, in the whole, the definition of the scope of an entity in a social network. Then, we have presented a possible investigation for extracting knowledge about entity scopes in social networks and we have implemented it on a real fragment of Twitter.

As for our future research efforts in this field, we plan: *(i)* to extend our approach in such a way as to make it able to operate on a generic social network, instead of on Twitter only; *(ii)* to empower our graph-based support model in such a way as to consider also tweet topics; *(iii)* to investigate how the knowledge about entity scope extracted by our approach can be applied in several contexts of social network research, for instance in information diffusion.

References

1. Acar, A., Muraki, Y.: Twitter for Crisis Communication: Lessons Learned from Japan’s Tsunami Disaster. *International Journal of Web Based Communities* 7(3), 392–402 (2011)
2. Benevenuto, F., Magno, G., Rodrigues, T., Almeida, V.: Detecting Spammers on Twitter. In: Proc. of the International Conference on Collaboration, Electronic Messaging, Anti-Abuse and Spam (CEAS 2010), Redmond, WA, USA (2010)
3. Buccafurri, F., Lax, G., Nocera, A., Ursino, D.: Discovering Links among Social Networks. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 467–482. Springer, Heidelberg (2012)
4. Cataldi, M., Di Caro, L., Schifanella, C.: Emerging Topic Detection on Twitter Based on Temporal and Social Terms Evaluation. In: Proc. of the International Workshop on Multimedia Data Mining (MDMKDD 2010), Washington, DC, USA, pp. 4–13. ACM (2010)
5. De Meo, P., Nocera, A., Terracina, G., Ursino, D.: Recommendation of similar users, resources and social networks in a Social Internetworking Scenario. *Information Sciences* 181(7), 1285–1305 (2011)
6. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proc. of the International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2001), San Francisco, CA, USA, pp. 57–66. ACM (2001)

7. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proc. of the International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2003), Washington, DC, USA, pp. 137–146. ACM (2003)
8. Ma, Z., Sun, A., Cong, G.: Will this #Hashtag be Popular Tomorrow? In: Proc. of the ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR 2012), Portland, OR, USA, pp. 1173–1174. ACM (2012)
9. Ma, Z., Sun, A., Cong, G.: On Predicting the Popularity of Newly Emerging Hashtags in Twitter. *Journal of the Association for Information Science and Technology* 64(7), 1399–1410 (2013)
10. Mendoza, M., Poblete, B., Castillo, C.: Twitter Under Crisis: Can We Trust What We RT? In: Proc. of the Workshop on Social Media Analytics (WSBA 2010), pp. 71–79. ACM, New York (2010)
11. Miller, Z., Dickinson, B., Deitrick, W., Hu, W., Wang, A.H.: Twitter Spammer Detection Using Data Stream Clustering. *Information Sciences* 260, 64–73 (2014)
12. Romero, D.M., Galuba, W., Asur, S., Huberman, B.A.: Influence and Passivity in Social Media. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 18–33. Springer, Heidelberg (2011)
13. Thomson, R., Ito, N., Suda, H., Lin, F., Liu, Y., Hayasaka, R., Isochi, R., Wang, Z.: Trusting Tweets: The Fukushima Disaster and Information Source Credibility on Twitter. In: Proc. of the International Conference on Information Systems for Crisis Response and Management (ISCRAM 2012), Vancouver, British Columbia, Canada (2012)
14. Wang, A.H.: Don't Follow Me - Spam Detection in Twitter. In: Proc. of the International Conference on Security and Cryptography (SECRYPT 2010), Athens, Greece, pp. 142–151. SciTePress (2010)
15. Wang, X., Wei, F., Liu, X., Zhou, M., Zhang, M.: Topic Sentiment Analysis in Twitter: A Graph-based Hashtag Sentiment Classification Approach. In: Proc. of the ACM International Conference on Information and Knowledge Management (CIKM 2011), Glasgow, Scotland, UK, pp. 1031–1040. ACM (2011)
16. Weng, J., Lim, E., Jiang, J., He, Q.: TwitterRank: Finding Topic-sensitive Influential Twitterers. In: Proc. of the ACM International Conference on Web Search and Data Mining (WSDM 2010), pp. 261–270. ACM, New York (2010)
17. Yang, J., Leskovec, J.: Patterns of Temporal Variation in Online Media. In: Proc. of the International Conference on Web Search and Web Data Mining (WSDM 2011), Hong Kong, China, pp. 177–186. ACM (2011)

Constructing Event Processing Systems of Layered and Heterogeneous Events with SPARQL

Mikko Rinne and Esko Nuutila

Department of Computer Science and Engineering,
Aalto University, School of Science, Finland
`{firstname.lastname}@aalto.fi`

Abstract. SPARQL was originally developed as a derivative of SQL to process queries over finite-length datasets encoded as RDF graphs. Processing of infinite data streams with SPARQL has been approached by using pre-processors dividing streams into finite-length windows based on either time or the number of incoming triples. Recent extensions to SPARQL can support interconnections of queries, enabling event processing applications to be constructed out of multiple incrementally processed collaborating SPARQL update rules. With more elaborate networks of queries it is possible to perform event processing on heterogeneous event formats without strict restrictions on the number of triples per event. Heterogeneous event support combined with the capability to synthesize new events enables the creation of layered event processing systems. In this paper we review the different types of complex event processing building blocks presented in literature and show their translations to SPARQL update rules through examples, supporting a modular and layered approach. The interconnected examples demonstrate the creation of an elaborate network of SPARQL update rules for solving event processing tasks.

Keywords: Complex event processing, SPARQL, heterogeneous events, stream processing.

1 Introduction

Complex event processing, as pioneered by David Luckham [15], takes a layered approach to creating event processing systems. Using the core definitions from the *Event Processing Glossary* [16], *an event* is broadly defined as “anything that happens, or is contemplated as happening” and *a complex event* is “an event that summarizes, represents, or denotes a set of other events”. Consequently the yardstick for complex event processing becomes the use of a layered model of event abstraction rather than the complexity of the underlying problem.

Event objects, representations of events for computer processing, may carry a variable number of parameters to describe an event with the accuracy required for a particular event processing application or set of applications [19]. In large

heterogeneous systems such as smart cities built from hardware and software components provided by multiple suppliers and operated by multiple independent private, corporate and public actors there can be a lot of variation between the data supplied by different sensors, requiring a flexible representation of event data. The need for flexibility is further backed up by concepts like *a composite event* [16], which is created by combining a set of simple or complex events, always encapsulating the component events from which the composite event is derived.

Semantic web standards RDF¹ and SPARQL² offer a good set of tools to cope with distributed heterogeneous environments. RDF is built with an open-world assumption and together with OWL offers tools to manage disjoint vocabularies. RDF graphs, with help from property paths and blank nodes, provide flexible means to encode event objects using variable numbers of elements and layers. TriG³ adds further structure by supporting the communication of RDF datasets in Turtle format. SPARQL, with support for federated queries, offers a natural way to enrich events with static background data available in the web e.g. as linked open data. Finally, the reasoning capabilities of different entailment regimes can be used to provide further semantic understanding of event data.

In [1] it is shown that an engine based on the Rete algorithm can incrementally process SPARQL queries offering competitive performance. The creation of an event processing application using multiple interconnected SPARQL queries has been described and tested in [18], while use-cases in the domain of semantic sensor systems have been further elaborated in [20]. In [8] a set of *event processing agents* to be used as building blocks in constructing event processing networks is proposed. The contribution of the present document is to demonstrate how the types of building blocks listed in [8] can be encoded in SPARQL to create an event processing network capable of handling heterogeneous events using standard unextended semantic web technologies.

The background of stream processing with SPARQL is explained in Section 2. Structured representations of heterogeneous events using RDF are reviewed in Section 3. SPARQL representations for different types of event processing agents are shown in Section 4. Conclusions and future areas of study are outlined in Section 5.

2 Stream Processing in SPARQL

The SPARQL query language, developed by W3C as the semantic web counterpart to SQL from the relational database world, was first released as a recommendation in January 2008. SPARQL is built to process queries over finite, possibly distributed datasets encoded in RDF. Event streams, such as the flows of measurements from sensors, are intrinsically infinite. Queries calculating aggregate values need to be computed over a finite partition of an event stream

¹ <http://www.w3.org/RDF/>

² <http://www.w3.org/TR/sparql11-query/>

³ <http://www.w3.org/TR/trig/>

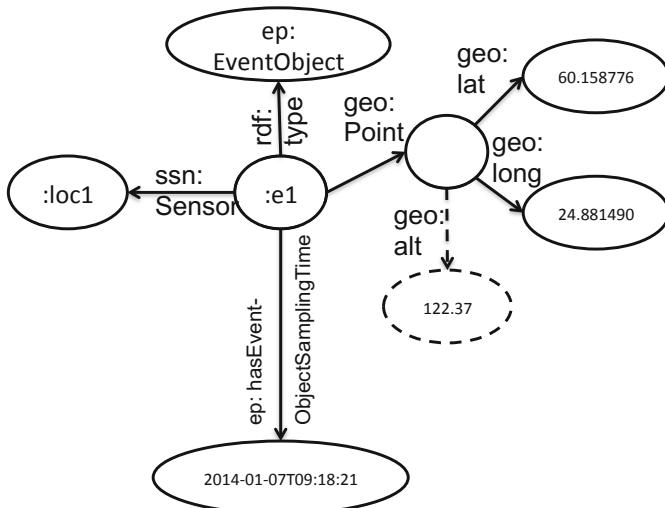


Fig. 1. Example Event for Sensor Location Reporting

to be solvable in finite time known as stream windows [16]. The first platform to demonstrate window-based stream processing in SPARQL was *C-SPARQL*⁴ [3,4,5]. In C-SPARQL RDF streams are built out of time-annotated triples. C-SPARQL provides a window mechanism, where the window size can be defined based on either time or the number of triples. Aggregation operators COUNT, MAX, MIN, SUM and AVG (later incorporated into SPARQL 1.1 Query specification) can be used to compute aggregate values from the windows. There is also a timestamp() function to access the timestamp of a variable. *CQELS*⁵ [14] was later introduced as another implementation of window-based streaming SPARQL.

Both C-SPARQL and CQELS assume repeated processing of queries over windows based on either time or number of triples. As pointed out in [18], this approach has some challenges:

- **No support for heterogeneous events:** Delimiting windows by time (with timestamps assigned to individual triples) or the number of triples carries a strong assumption that each event is represented by a single triple, an approach sometimes referred to as data stream processing [13]. Event objects consisting of a variable number of triples would be split across window borders (or merged, if the timestamps are identical). Objects consisting of a static number of triples could be windowed based on the number of triples, but this approach easily gets out-of-sync due to a missing triple.

⁴ <http://streamreasoning.org/download>

⁵ <http://code.google.com/p/cqels/>

- **Challenges in window dimensioning:** When a query involves multiple events in an event stream, the window has to be large enough to capture all the required events. To avoid missing event patterns due to window borders, the windows have to overlap, resulting in duplicate processing and duplicate detections. Faster window repetition helps to decrease detection delay, but leads to more duplicate processing and detections. In an event processing system these conflicting requirements lead to unwanted compromises between duplicate processing, duplicate detections and notification delay.
- **Wasted resources:** Queries are processed repeatedly over the defined windows, even when there are no new matching events.

EP-SPARQL [2] is a streaming environment focusing on the detection of RDF triples in a specific temporal order. The published examples also support heterogeneous event formats, create aggregation over sliding windows using subqueries and expressions, and layer events by constructing new streams from the results of queries. The prolog-based ETALIS⁶ incorporates more functionality than the EP-SPARQL front-end.

Another approach to stream processing is to incrementally and asynchronously process each incoming event against a pre-defined set of queries. Instead of windows based on time or number of triples, all processing is based on events, which contribute new input to the queries. When all the required inputs of a query are present, the result is immediately available. Algorithms familiar from production rule systems, such as Rete [9,10], can be adapted to use SPARQL to describe the rules.

Streaming SPARQL was first presented in [11] using a network highly similar to Rete. Komazec and Cerri [13] apply SPARQL queries to RDF data using an extended Rete-algorithm in a system called *Sparkwave*⁷. Their focus is on supporting selected RDF and RDFS inference rules through the use of a pre-processing ϵ network and fast processing of data streams consisting of individual triples instead of multi-triple events. Rete has also been used for processing SPARQL queries by Depena and Miranker in [7,17], where the focus is on nested access of dereferenced URIs, exploiting the constructive analogy with forward-chaining production systems.

A SPARQL-based event stream processing platform denoted INSTANS⁸ has been developed in our research group. The query language is SPARQL 1.1, implementing also selected properties of SPARQL 1.1 Update, such as *INSERT* and *DELETE*. In [18] we have described the creation of event processing applications using interconnected SPARQL update rules. The concept of collaborating SPARQL queries for stream processing is discussed also in [22], with Teymourian et al preferring a backward-chaining algorithm over the forward-chaining type used in Rete.

⁶ <http://code.google.com/p/etalis/>

⁷ <https://github.com/skomazec/Sparkweave>

⁸ Incremental eNginE for STANDING Sparql, <http://instans.org/>

3 Event Objects in RDF

In *data stream processing* it is assumed that each triple carries a standalone event. This can typically be a single reading from a temperature sensor or the license plate of a car from a tollgate. To minimize redundant data transfer and processing in the system, periodic measurements should be filtered as close to the source as possible, elevating to a higher level of abstraction, where only sensor readings falling outside of previously defined reporting thresholds or otherwise unexpected results are forwarded upstream as events. Such event objects typically carry more information than a single sensor reading and timestamp. RDF is better motivated as the data format for heterogeneous asynchronously triggered events than frequent periodic measurements of a single parameter, for which there are more compact representations. Different sensors, or even dif-

```

BASE <http://instans.org/>
PREFIX : <http://instans.org/default#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX ep: <http://www.ontologydesignpatterns.org/cp/owl/eventprocessing.owl#>
<Eve1> {
    <Eve1> a ep:EventObject;
    ssn:Sensor :loc1;
    geo:Point [ geo:lat 60.158776 ; geo:long 24.881490 ; ] ;
    ep:hasEventObjectSamplingTime "2014-01-07T09:18:21"^^xsd:dateTime }
```

Fig. 2. Prefixes and an example event

ferent driver software versions of the same sensor, may include different types of auxiliary data in addition to the base parameters used by our event processing application. Also in [8] the assumed structure of an event includes a variable-length header, which the event processing system can process, and a body, which is transported as unstructured payload. These structural requirements have been addressed by the *event processing ontology*⁹ [19]. To save space, the event objects in this paper are simple events without separate header and body parts, but the extensions to support header, body and composite events from the event processing ontology can be used to extend the examples presented herein as demonstrated in [19].

A sample event format, for the purpose of updating the location of a sensor, is illustrated in Figure 1. The dotted altitude field exemplifies an optional field, which may not be included by all sensors. The turtle serialization encapsulated in TriG is shown in Figure 2. Even though RDF triples can be used to build graphs of infinitely complex structures, there is no clear and reliable way to express the boundaries of a “data record” such as an event object, as pointed out in [12]. This may cause problems when event objects in a stream contain optional elements.

⁹ <http://ontologydesignpatterns.org/wiki/Submissions:EventProcessing>

A query often produces different results depending on whether some optional elements are present, but there is no rule for how long the optional triples should be waited for in the case of an infinite stream. INSTANS can be parametrized to jointly process a “block” of data, which in the case of Turtle jointly processes all consecutive triples connected by a common subject or blank nodes according to rule #6 (“triples”) of the Turtle grammar¹⁰. However, something as simple as sending the latitude and longitude coordinates of Figure 2 before the rest of the event may cause problems: A first block terminates already after the coordinates which, depending on the associated queries, may trigger results different from the case where the coordinates are sent after the event. The issue is further emphasized in a rule-based system like INSTANS, where the user has no explicit control over the order in which simultaneously matching rules are executed and therefore the order in which triples are output.

TriG [6] defines a way of communicating datasets as Turtle by encapsulating graphs. With this approach each event object can be defined as a graph, and each graph processed as a block with no ambiguity. This has clear benefits in matching incoming event objects having optional or unknown content. In [19] unknown content is matched using nested OPTIONAL clauses tracking triples connected by blank nodes. With TriG the situation is greatly simplified, because the entire incoming event object - independent of structural complexity or order of triples - can be matched simply with “?s ?p ?o”, as e.g. in Figure 5. The tradeoff is that some of the granularity in using named graphs is lost: Each event object is encapsulated into a named graph and since SPARQL doesn’t offer regular expression matching for graphs in one step, the incoming event objects have to be matched to variables and filtered in two separate steps, as shown in the examples in Section 4. As TriG is a new specification, there is currently no specified way to construct TriG output in SPARQL. INSTANS implements a small extension to enable construction of graphs, in TriG-format, using the regular CONSTRUCT operator.

4 Event Processing Agents in SPARQL

The basic building blocks of an *Event Processing Network* (EPN) are shown in Figure 3. *Event Processing Agents* (EPA, 3.) operate on events, which are transported between EPA:s over *Event Channels* (2.). Two special types of EPA:s are shown in the figure: *Event Producer* (1.), which is a source of events and has no inputs in this EPN and *Event Consumer*, which is an event sink and has no outputs in this EPN. It should be noted, however, that in a system with multiple EPN:s the Event Consumer of one EPN can typically be an Event Producer in another. When translated to SPARQL, EPA:s are implemented as SPARQL Update rules (or networks of rules). Event Channels can be modelled as named graphs, providing isolated connection points for event object transfer between EPA:s. The types of event processing agents from [8] are shown in Figure 4. In the following subsections the SPARQL implementation of each type, complemented

¹⁰ <http://www.w3.org/TR/turtle/#sec-grammar>

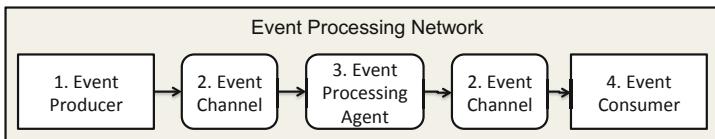


Fig. 3. Building Blocks of an Event Processing Network

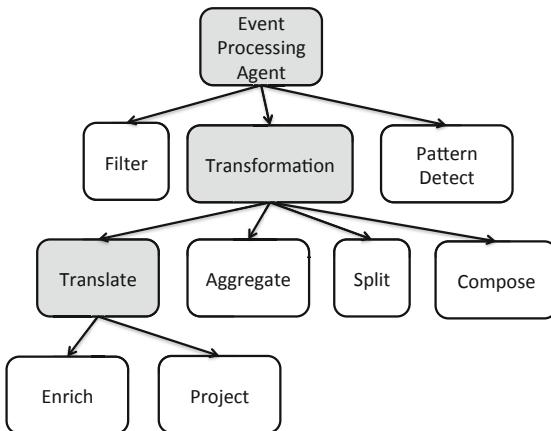


Fig. 4. Types of Event Processing Agents (from [8])

by the set of filter types from [21], are examined to show the implementations as SPARQL update rules.

4.1 Filters

Filters select interesting events or eliminate uninteresting ones. A forwarding decision is made based on a test or set of tests. In a stateless filter the decision is fully based on the current event object. An example (**EPA 1**) of a stateless filter to pass through all events generated during business hours is in Figure 5. Since only the time attribute is needed for filtering, it is the only explicitly matched object. Otherwise the entire incoming event object is matched with “?s ?p ?o” and copied in case it passes the filter. The `str(<>)` function produces the BASE URI as a string. It is needed because the graph and event names are prefixed with the BASE URI when read from a file. The `BIND`-statement is used to prefix the output graph name with “Poststateless”, separating it from incoming event objects and thereby creating a new event channel. A stateful filter utilizes information external to the incoming event, e.g. information based on previously processed events or an external source. A stateful filter (**EPA 2**) to pass through maximally one location update for each even hour is shown in Figure 6. This filter, implemented as three SPARQL update rules, demonstrates how a graph is

```

INSERT { GRAPH ?poststateless { ?s ?p ?o } }
WHERE { GRAPH ?g { FILTER (strStarts(str(?g),concat(str(<>),"Eve")))
    ?s ?p ?o . ?event ep:hasEventObjectSamplingTime ?time }
    FILTER ( ( HOURS(?time) > 8 ) && ( HOURS(?time) < 17 ) )
    BIND (IRI(concat("Poststateless-",strAfter(str(?event),str(<>)))) AS ?poststateless) }

```

Fig. 5. EPA 1: A stateless filter passing through events generated during business hours

```

# EPA2-1 Initialize
INSERT DATA { GRAPH <memory> { :stateful :hour -1 ; :passEvent [] } };
# EPA2-2 Update memory, mark passing events
DELETE { GRAPH <memory> { :stateful :hour ?prevhour ;
                           :passEvent ?prevPass } }
INSERT { GRAPH <memory> { :stateful :hour ?hour ;
                           :passEvent ?event } }
WHERE { GRAPH ?g { FILTER ( strStarts(str(?g),"Poststateless-Eve") )
    ?event ep:hasEventObjectSamplingTime ?time }
    GRAPH <memory> { :stateful :hour ?prevhour ;
                           :passEvent ?prevPass } }
BIND ( HOURS(?time) as ?hour)
FILTER ( ?hour != ?prevhour ) } ;
# EPA2-3 Pass events marked by EPA2-2
INSERT { GRAPH ?poststateful { ?s ?p ?o } }
WHERE {
    GRAPH <memory> { :stateful :passEvent ?event }
    GRAPH ?g { FILTER ( strStarts(str(?g),"Poststateless-Eve") )
        ?event ep:hasEventObjectSamplingTime ?time . ?s ?p ?o }
    BIND (IRI(concat("Poststateful-",strAfter(str(?event),str(<>)))) AS ?poststateful) } ;

```

Fig. 6. EPA 2: A stateful filter passing through one event per even hour

used as memory to save the hour of the previous event for comparison. Explicit memory initialization in EPA2-1 is used to avoid OPTIONAL statements in the other queries. It can be observed that implementing a stateful filter with a streaming SPARQL system based on window repetition would be practically impossible in scenarios where one window should incorporate multiple events. Because all the events in a window are processed jointly as a batch, results of processing one event cannot influence the processing of the next one. Therefore stateful filters such as the example in Figure 6 typically require a continuous engine, which processes each event as it arrives in a stream.

In principle a SPARQL-based event processing network can support all types of filters, which can be computed with the available arithmetic operators, taking into account that multiple filter stages can be chained like any other EPA:s. One clear limitation is the absence of square root from SPARQL, which causes the calculation of geographical distances to be usually processed with extension functions. As a collection of filter types Table 1 shows the filters listed in

[21] as “input specifiers” together with the corresponding SPARQL FILTER statements. The first four filters are clearly stateless. The last one, *change*, does stateful comparison, but it would also be simpler to implement as a combination of an *aggregate* block (EPA6 below) to compute a sliding aggregate value (e.g. min, max, average) over the n previous readings and compare the output of that block to the latest value using a stateless filter.

Table 1. SPARQL implementations of the filter types listed in [21]

Description:	SPARQL filter
Equal: Check for equality	FILTER (?value1 = ?value2)
About: Equality within a defined tolerance value	FILTER (abs(?value1 - ?value2) < tolerance)
Area: Interval between two values	FILTER ((?value > lower_bound) && (?value < upper_bound))
Greater / Less: Check if the value is greater or less than a defined value.	FILTER (?value > limit) or FILTER (?value < limit)
Change: Tracks changes compared to n previous readings. “Decrease” and “Increase” can be used to specify the direction of change.	Two rules recommended: An aggregate to compute “n previous readings” and a stateless filter to compare.

4.2 Transformation

Transformation agents modify the content of received event objects. Subtypes are:

- **Translate:** Operate on each event object independently of preceding or subsequent objects using a single in - single out model. A special case is *Enrich*, which attaches additional information to an event object. SPARQL is particularly suitable for this purpose, since any data from a SPARQL endpoint can be queried as a federated query. An example is given in **EPA 3** (Figure 7), where our location events are enriched with preferred geographical labels from *FactForge*¹¹.
- **Project:** Remove information from the incoming event. A simple example of this (**EPA 4**) would be the removal of the recently added location names. As this is a simple copy operation of an incoming event leaving out the unwanted parts, the SPARQL listing is omitted as trivial.
- **Split:** Split a single incoming event into multiple outgoing events. In **EPA 5** (Figure 8) incoming events are split to two channels.
- **Aggregate:** Output a function of incoming events in a multiple-in single-out model. Typical examples are count, min, max, average and sum. The example in **EPA 6** (Figure 9) counts incoming events per hour. It employs five rules to count the aggregate number of events per hour. EPA6-1 initializes the

¹¹ <http://factforge.net>

memory graph. EPA6-2 compacts SPARQL code lines by extracting the hour of the incoming event, used by all three subsequent queries. EPA6-3 increases the event counter, 6-4 outputs the event counts when the hour changes and 6-5 resets the counter. Since the output is triggered by an object of the next hour, the last counter result is never produced. This may not be critical in an infinite stream setting, but for recorded streams the final output can be triggered e.g. by a special end marker triple at the end of the file. Alternatively the count could be updated after every event object.

- **Compose:** Combine two or more incoming streams to a single output stream. An example is given as **EPA 7** in Figure 10, a very straightforward reversal of EPA5 with the contents of the *INSERT* and *WHERE* clauses reversed.

Many streaming SPARQL platforms [3,14] implement proprietary SPARQL extensions for computation of aggregate values over time windows. These tailor-made solutions result in compact queries, but currently suffer from problems in supporting heterogeneous events due to the window size definitions outlined in Section 2 and different timecodes in streams. There is currently no specified way to use e.g. ep:hasEventObjectSamplingTime from the sample events of this paper as the time base for these window operators, but W3C RDF

```
PREFIX omgeo: <http://www.ontotext.com/owlim/geo#>
PREFIX ff: <http://factforge.net/>
INSERT { GRAPH ?translated { ?event :locationName ?label . ?s ?p ?o } }
WHERE { GRAPH ?g { FILTER ( strStarts(str(?g),"Poststateful-Eve") )
    ?s ?p ?o . ?event a ep:EventObject .
    SERVICE <http://factforge.net/sparql> { # Retrieve location label
        ?location omgeo:nearby(?lat ?long "1km"); ff:preferredLabel ?label }
    BIND (IRI(concat("Translated-",strAfter(str(?event),str(<>)))) AS ?translated) } }
```

Fig. 7. EPA 3: Enriches incoming events with location labels

```
INSERT { GRAPH ?geograph { ?event a ep:EventObject ;
    ep:hasEventObjectSamplingTime ?time ;
    geo:Point [ geo:lat ?lat ; geo:long ?long ; ] }
GRAPH ?sensorgraph { ?event a ep:EventObject ; ssn:Sensor ?sensor ;
    ep:hasEventObjectSamplingTime ?time }
WHERE { GRAPH ?g { FILTER ( strStarts(str(?g),"Poststateful-Eve") )
    ?event a ep:EventObject ; ssn:Sensor ?sensor ;
    geo:Point [ geo:lat ?lat ; geo:long ?long ; ] ;
    ep:hasEventObjectSamplingTime ?time }
BIND (IRI(concat("GeoEvents-",strAfter(str(?event),str(<>)))) AS ?geograph)
BIND (IRI(concat("SensorEvents-",strAfter(str(?event),str(<>)))) AS ?sensorgraph) }
```

Fig. 8. EPA 5: Event objects split to two channels

```

# EPA6-1 Initialize
INSERT DATA { GRAPH <memory> { :aggrHour :hour -1 ; :counter -1 } } ;
# EPA6-2 Extract incoming hour
INSERT { GRAPH <memory> { ?event :hasAggrHour ?hour } }
WHERE { GRAPH ?g { FILTER ( strStarts(str(?g),concat(str(<>),"Eve")) ) 
    ?event ep:EventObjectSamplingTime ?time } }
BIND ( HOURS(?time) as ?hour ) ;
# EPA6-3 Increase event counter
DELETE { GRAPH <memory> { ?x :counter ?oldcount . ?event :hasAggrHour ?hour } }
INSERT { GRAPH <memory> { ?x :counter ?newcount } }
WHERE { GRAPH <memory> { ?x :hour ?memhour ;
    :counter ?oldcount . ?event :hasAggrHour ?hour } }
FILTER(?hour = ?memhour)
BIND ( ?oldcount + 1 AS ?newcount ) ;
# EPA6-4 Output event counts
CONSTRUCT { GRAPH <eventcounts> { # Insert counter event
    [] a ep:EventObject ; :eventType :eventCount ;
        :hour ?memhour ; :count ?count } }
WHERE { GRAPH <memory> { ?x :hour ?memhour ; :counter ?count .
    ?event :hasAggrHour ?hour }
    FILTER ( ?memhour != -1 && ?memhour != ?hour ) } ;
# EPA6-5 Reset memory
DELETE { GRAPH <memory> { ?x :hour ?memhour ; :counter ?count } }
INSERT { GRAPH <memory> { [] :hour ?hour ; :counter 0 } }
WHERE { GRAPH <memory> { ?x :hour ?memhour ; :counter ?count .
    ?event :hasAggrHour ?hour }
    FILTER ( ?memhour != ?hour ) } ;

```

Fig. 9. EPA 6: Produce aggregate events by counting the number of incoming events per hour

```

INSERT { GRAPH ?combined {
    ?event a ep:EventObject ; ssn:Sensor ?sensor ;
        geo:Point [ geo:lat ?lat ; geo:long ?long ; ] ;
        ep:hasEventObjectSamplingTime ?time } }
WHERE { GRAPH ?g1 { FILTER(strStarts(str(?g1),"GeoEvents-"))
    ?event a ep:EventObject ;
        geo:Point [ geo:lat ?lat ; geo:long ?long ; ] }
GRAPH ?g2 { FILTER(strStarts(str(?g2),"SensorEvents-"))
    ?event a ep:EventObject ; ssn:Sensor ?sensor ;
        ep:hasEventObjectSamplingTime ?time }
BIND (IRI(concat("Combined-",strAfter(str(?event),str(<>)))) AS ?combined) }

```

Fig. 10. EPA 7: Combine the streams, which were earlier split to GeoEvents and SensorEvents

Stream Processing Community Group¹² is working on defining common models. With manually generated aggregate calculation rules windowing can be based on any parameter (e.g. location), not just time, from the incoming RDF stream.

¹² <http://www.w3.org/community/rsp/>

Multiple aggregate values can also be computed jointly. In the example above the sum of latitude and longitude coordinates could also be recorded, after which it would be easy to obtain average locations.

As INSTANS operates asynchronously, windowing based on a real-time clock would require the use of *timed events* (“pulse” event objects triggered based on a clock, in this case to trigger aggregate calculation). In many practical cases using timestamps from the incoming stream is a better approach, because it works on both live and recorded streams and produces more predictable and repeatable results than referencing the clock of the computer running the stream processing platform. The downside is that the absence of an out-of-window trigger may leave the current result waiting infinitely.

4.3 Pattern Detect

Pattern detect agents are searching for patterns in incoming events. They may either describe the pattern, pass through qualifying patterns of incoming event objects or both. Simple patterns could be detected e.g. with slightly augmented

```
# EPA8-Transform Initialize memory with 1st incoming point from ?sensor
INSERT { GRAPH <memory> { [] a :transformPrevPoint ; ssn:Sensor ?sensor ;
    geo:Point [ geo:lat ?lat ; geo:long ?long ; ] } }
WHERE { GRAPH ?g { FILTER (strStarts(str(?g),concat(str(<>),"Eve")))
    ?event a ep:EventObject ; ssn:Sensor ?sensor ;
    geo:Point [ geo:lat ?lat ; geo:long ?long ; ] }
    FILTER NOT EXISTS {
        GRAPH <memory> { ?x a :transformPrevPoint ; ssn:Sensor ?sensor } } } }

# EPA8-Transform subsequent points to compass directions
DELETE { GRAPH <memory> { # Delete previous point
    ?mem geo:Point ?blnk . ?blnk geo:lat ?prevlat ; geo:long ?prevlong } }
INSERT { GRAPH <memory> { # Write new point
    ?mem geo:Point [ geo:lat ?lat ; geo:long ?long ; ] }
    GRAPH <directions> { # Output direction
        ?event a ep:EventObject ; ssn:Sensor ?sensor ;
        ep:hasEventObjectSamplingTime ?time ; :direction ?dir } }
WHERE { GRAPH ?g { FILTER (strStarts(str(?g),concat(str(<>),"Eve")))
    ?event a ep:EventObject ; ssn:Sensor ?sensor ;
    ep:hasEventObjectSamplingTime ?time ;
    geo:Point [ geo:lat ?lat ; geo:long ?long ; ] . }
    GRAPH <memory> { # Retrieve previous point
        ?mem a :transformPrevPoint ; ssn:Sensor ?sensor ;
        geo:Point ?blnk . ?blnk geo:lat ?prevlat ; geo:long ?prevlong } }
BIND ( IF ( ?lat > ?prevlat, "N", "") as ?latdir1) # Test for directions
BIND ( IF ( ?lat < ?prevlat, "S", "") as ?latdir2)
BIND ( IF ( ?long > ?prevlong, "E", "") as ?longdir1)
BIND ( IF ( ?long < ?prevlong, "W", "") as ?longdir2)
BIND ( concat(?latdir1, ?latdir2, ?longdir1, ?longdir2) as ?tdir )
BIND ( IF ( !bound(?tdir) || strlen(?tdir)=0, "0", ?tdir ) as ?dir ) }
```

Fig. 11. Transform: Generate direction events out of subsequent locations

stateful filters. A **pattern detect example (EPA 8)** to detect a pattern of movement directions from consecutive location updates is shown here as an example:

1. **Transform location updates to a stream of directions:** Auxiliary query creating a stream of compass events (e.g. “NW”) out of an incoming stream of location updates, Figure 11.
2. **Advance pattern index with positive match:** Compare incoming direction events with a pattern in a graph, advance index when they match, Figure 12.
3. **Reset pattern index when not matching:** If a direction event does not match the next item in the pattern, reset index in memory graph, Figure 13.
4. **Detect completed pattern:** When the pattern is complete, indicate detection and reset the index, Figure 14.

It is worth noting that the WHERE-clauses in the two queries of Figure 14 are completely identical. The only reason why they cannot be merged is that SPARQL currently does not allow DELETE and CONSTRUCT in the same query.

```
# EPA8-Pattern-Initialize memory with 1st direction entry from sensor
INSERT { GRAPH <memory> { # Initialize a new entry to memory
    [] a :patternIndexEntry ; ssn:Sensor ?sensor ;
        :basedOnEvent 0 ; :patternIndex 0 } }
WHERE { GRAPH <directions> { ?event a ep:EventObject ; ssn:Sensor ?sensor } }
      FILTER NOT EXISTS { GRAPH <memory> {
          ?mem a :patternIndexEntry ; ssn:Sensor ?sensor } } } ;
# EPA8-Pattern-Advance index when direction matches with pattern
DELETE { GRAPH <memory> { # Clear old index from memory
    ?mem :basedOnEvent ?oldEvent ; :patternIndex ?oldIndex } }
INSERT { GRAPH <memory> { # Write new index to memory
    ?mem :basedOnEvent ?event ; :patternIndex ?newIndex } }
WHERE { GRAPH <directions> { ?event a ep:EventObject ;
                                ssn:Sensor ?sensor ; :direction ?dir } }
      FILTER ( ?dir != "0" )
GRAPH <memory> { # Retrieve index from memory
    ?mem a :patternIndexEntry ; ssn:Sensor ?sensor ;
        :basedOnEvent ?oldEvent ; :patternIndex ?oldIndex }
      FILTER ( !sameTerm(?event,?oldEvent) ) # Only once per event
GRAPH <pattern> { # Retrieve pattern index and value
    ?pattern :index ?ptIndex ; :value ?ptValue }
      BIND ( ?oldIndex+1 as ?newIndex )
      FILTER ( (?dir = ?ptValue) && (?newIndex = ?ptIndex) ) }
```

Fig. 12. EPA 8-1: Advance an index if the incoming direction matches with a pre-defined pattern

```

DELETE { GRAPH <memory> { # Remove index from memory
    ?mem a :patternIndexEntry ; ssn:Sensor ?sensor ;
          :basedOnEvent ?oldEvent ; :patternIndex ?oldIndex } }
WHERE { GRAPH <directions> { # Match directional input
    ?event a ep:EventObject ; ssn:Sensor ?sensor ; :direction ?dir }
GRAPH <memory> { # Retrieve index from memory
    ?mem a :patternIndexEntry ; ssn:Sensor ?sensor ;
          :basedOnEvent ?oldEvent ; :patternIndex ?oldIndex }
FILTER ( !sameTerm(?event,?oldEvent) ) # Must be different events
GRAPH <pattern> { ?pattern :index ?ptIndex ; :value ?ptValue }
FILTER ( (?dir != ?ptValue) && (?oldIndex+1 = ?ptIndex) && (?oldIndex>0) ) }

```

Fig. 13. EPA 8-2: Reset index if the latest direction does not match with the pattern

```

# EPA8-Pattern-Reset index when pattern is complete
DELETE { GRAPH <memory> { # Remove last index from memory
    ?mem a :patternIndexEntry ; ssn:Sensor ?sensor ;
          :basedOnEvent ?oldEvent ; :patternIndex ?oldIndex } }
WHERE { GRAPH <pattern> { :pattern :length ?length }
GRAPH <memory> {
    ?mem a :patternIndexEntry ; ssn:Sensor ?sensor ;
          :basedOnEvent ?oldEvent ; :patternIndex ?oldIndex }
FILTER ( ?oldIndex = ?length ) } ;
# EPA8-Pattern-Output result
CONSTRUCT { GRAPH <PatternDetect> {
    [] a ep:EventObject ; :patterndetected "Pattern detected!" ;
        ssn:Sensor ?sensor ; :lastEvent ?oldEvent } }
WHERE { GRAPH <pattern> { :pattern :length ?length }
GRAPH <memory> {
    ?mem a :patternIndexEntry ; ssn:Sensor ?sensor ;
          :basedOnEvent ?oldEvent ; :patternIndex ?oldIndex }
FILTER ( ?oldIndex = ?length ) } ;

```

Fig. 14. EPA 8-3: Output result when the designated pattern is complete. Reset pattern index.

5 Conclusions

The benefits of using SPARQL for event processing include all the benefits of semantic web in general, e.g. agreed specifications, conceptual compatibility through shared ontologies, compatibility with the current tool base, support for loosely coupled heterogeneous systems, straightforward connectivity to linked open data for enriching event information and a solid base of inference mechanisms to enhance reasoning over events. The principle of processing heterogeneous events using a network of SPARQL queries and update rules was first presented in [18] through a practical event processing example solved by three interconnected rules and a result output query. This paper takes a more systematic approach by showing working examples of every type of event processing agent found in [8]. The SPARQL implementation of the “input modifiers” (filter

types) mentioned in [21] is also reviewed. Building on top of the principles introduced in [19], the benefits of TriG input over plain Turtle in encapsulating events objects are explained and demonstrated. While the examples in this paper have been prepared manually, it would also be possible to create end-user tools to synthesize the appropriate rules and queries.

Based on the examples it is observed that every type of event processing agent found in the referenced literature can be expressed using a SPARQL update rule or network of rules. While the examples do not extend to prove that every event processing task would have a corresponding solution in SPARQL, they serve to demonstrate a systematic approach up to a level of complexity, which would be sufficient for many real-life scenarios. SPARQL is observed to be capable of serving as the foundation for event processing systems. However, limitations especially in arithmetic operators are frequently forcing the inclusion of proprietary library functions, e.g. for the calculation of distance between two geographical points.

The experiment has also revealed in SPARQL 1.1 some structural shortcomings, which could be addressed in future releases of the specification set:

1. *Generating dataset output from a query:* Even though TriG is specified as a format to convey datasets and SPARQL can query TriG input, there is currently no way to produce dataset output with a query. We have added to CONSTRUCT the capability to generate graphs as TriG output and have observed no negative impacts either to the coherence of the syntax or the INSTANS implementation.
2. *Combination of output and update:* As we saw in Figure 14, the SPARQL code could be made more compact by allowing multiple result processing operators (e.g. CONSTRUCT and INSERT) in the same query. Even more importantly, it is currently very difficult to chain any operation to take place after output processing, because SELECT or CONSTRUCT queries cannot produce any changes detectable by other queries.

In EPA6 (Figure 9) some codelines were reduced by adding an extra query to extract a parameter used by three other queries. INSTANS automatically merges identical parts of queries in the internal Rete representation. Therefore this reduction is only significant in terms of SPARQL text; the compiled rule processing network is the same. Due to limits in the capabilities of a single query, similar parts often need to be used in multiple queries. A macro mechanism would reduce the need to replicate the same SPARQL code to multiple places, compacting the source code and reducing opportunities for inconsistency.

The individual example rules presented in this paper connect to each other, forming the event processing network shown in Figure 15. This size of network would already be capable of solving quite complex event processing tasks. The complete set of rules is available in executable form in the INSTANS github repository¹³. Memory management was not discussed in detail in this document, but the complete query set in the repository avoids garbage buildup by two means:

¹³ <https://github.com/aaltodsg/instans/tree/master/tests/input/CEP2SPARQL>

1. An operational policy of INSTANS removes input triples after all the rules, which match with a given input, have been processed.
2. Explicit “cleanup”-queries delete old events from the channels between queries.

While putting all of these rules into a single Rete engine serves to demonstrate that complex networks can be created, it is not the most efficient solution. No EPA shown here is exchanging any other information than event objects with other EPA:s, in which case it would be better to run each EPA in a separate instance of INSTANS. This modularizes the approach and ensures that there will not be any unwanted side-effects due to unplanned matching of rules between different agents.

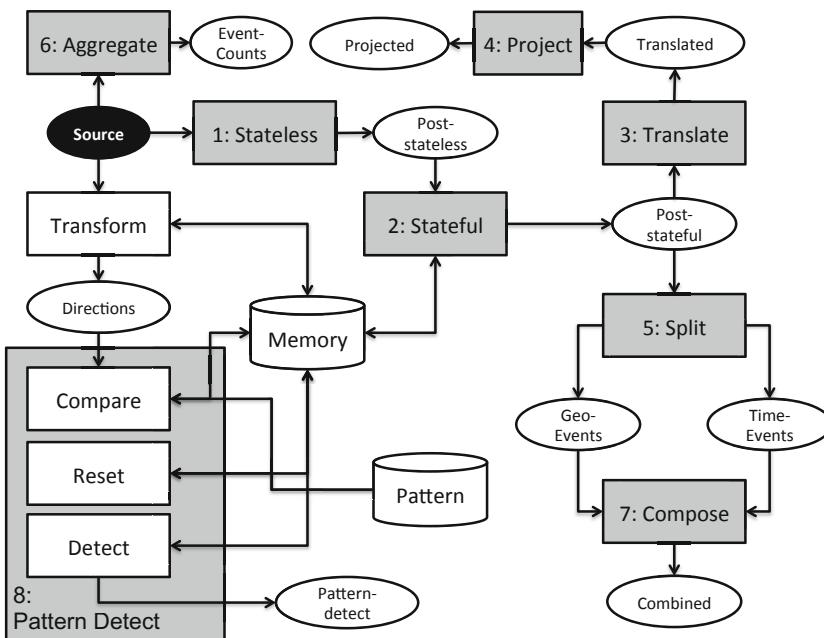


Fig. 15. The example event processing network created in the paper

The challenges of applying synchronous query repetition over windows in event processing tasks were discussed in [18]. In this document it was additionally observed that the creation of stateful filters has further challenges in windowed environments because conventional SPARQL matches graph patterns over the complete addressable dataset, in this case window, before acting on the results. Unless it can be guaranteed that a window contains precisely one event object (thereby precluding cases using multiple event objects from the same stream as input), the results of processing one event cannot impact the processing of the

next. The results of stateful filtering can also be sensitive to repeated processing, causing symptoms with overlapping (non-tumbling) windows.

After translation of event processing tasks to SPARQL, the next step will be to describe the translation of SPARQL to a query network. Not all aspects of SPARQL are applicable to continuous event-triggered processing, and in some cases the semantics need to be adjusted. The aspects of synchronized aggregate calculation and processing of timed events in general also need to be elaborated further. While the synchronized computation of aggregate values is not seen as a key aspect of an asynchronous event processing platform, some reporting tasks and cleaning of noisy sensor input may require aggregate calculations. Timed events are also essential in detecting missing events and need to be addressed in more detail in future work.

Acknowledgments. This work has been carried out in Spaceify, SPIRE and TrafficSense projects funded by European Commission through the SSRA (Smart Space Research and Applications) activity of EIT ICT Labs¹⁴, Tekes and Aalto University.

References

1. Abdullah, H., Rinne, M., Törmä, S., Nuutila, E.: Efficient matching of SPARQL subscriptions using Rete. In: Proceedings of the 27th Symposium on Applied Computing, Riva del Garda, Italy (March 2012)
2. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: Proceedings of the 20th International Conference on World Wide Web (WWW 2011), pp. 635–644. ACM (2011)
3. Barbieri, D.F., Braga, D., Ceri, S., Grossniklaus, M.: An execution environment for C-SPARQL queries. In: Proceedings of the 13th International Conference on Extending Database Technology, EDBT 2010, Lausanne, Switzerland, p. 441 (2010)
4. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: C-SPARQL: A Continuous query language for RDF data streams. International Journal of Semantic Computing 04, 3 (2010)
5. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Querying RDF streams with C-SPARQL. ACM SIGMOD Record 39, 20 (2010)
6. Bizer, C., Cyganiak, R.: RDF 1.1 TriG, <http://www.w3.org/TR/trig/>
7. Depena, R.K.: Diamond: A Rete-Match Linked Data SPARQL Environment (M.Sc. Thesis). Ph.D. thesis, University of Texas at Austin (2010)
8. Etzion, O., Niblett, P., Luckham, D.: Event Processing in Action. Manning Publications (July 2010)
9. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence 19(1), 17–37 (1982)
10. Forgy, C.L.: On the efficient implementation of production systems. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA (1979), aAI7919143
11. Groppe, S., Groppe, J., Kukulenz, D., Linnemann, V.: A SPARQL Engine for Streaming RDF Data. In: Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, pp. 167–174. IEEE (December 2007)

¹⁴ <http://eit.ictlabs.eu/ict-labs/thematic-action-lines/smart-spaces/>

12. Keskisärkkä, R., Blomqvist, E.: Event Object Boundaries in RDF Streams - A Position Paper. In: OrdRing 2013 - 2nd International Workshop on Ordering and Reasoning. CEUR Workshop Proceedings (2013)
13. Komazec, S., Cerri, D.: Towards Efficient Schema-Enhanced Pattern Matching over RDF Data Streams. In: 10th ISWC 2011, Bonn, Germany. Springer (2011)
14. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
15. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems, 1st edn. Addison-Wesley Professional (May 2002)
16. Luckham, D., Schulte, R.: Event Processing Glossary Version 2.0 (July 2011), <http://www.complexevents.com/>
17. Miranker, D.P., Depena, R.K., Hyunjoon, J., Carlos, R., Sequeda, J.F.: Diamond: A SPARQL Query Engine, for Linked Data Based on the Rete Match. In: Gueret, C., Sharffle, F., Ienco, D., Villata, S. (eds.) 1st International Workshop on Artificial Intelligence Meets the Web of Data (ECAI 2012), Montpellier, pp. 12–17 (2012)
18. Rinne, M., Abdullah, H., Törmä, S., Nuutila, E.: Processing Heterogeneous RDF Events with Standing SPARQL Update Rules. In: Meersman, R., et al. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 797–806. Springer, Heidelberg (2012)
19. Rinne, M., Blomqvist, E., Keskisärkkä, R., Nuutila, E.: Event Processing in RDF. In: Proceedings of WOP 2013. CEUR Workshop Proceedings, p. 13 (2013)
20. Rinne, M., Törmä, S., Nuutila, E.: SPARQL-Based Applications for RDF-Encoded Sensor Data. In: 5th International Workshop on Semantic Sensor Networks (2012)
21. Taylor, K., Leidinger, L.: Ontology-Driven Complex Event Processing in Heterogeneous Sensor Networks. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 285–299. Springer, Heidelberg (2011)
22. Teymourian, K., Rohde, M., Paschke, A.: Fusion of background knowledge and streams of events. In: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, DEBS 2012, pp. 302–313. ACM Press, New York (2012)

On Efficient Processing of Linked Stream Data

Omran Saleh and Kai-Uwe Sattler

Department of Computer Science and Automation,
Technische Universität Ilmenau, Germany
`{first.last}@tu-ilmenau.de`

Abstract. Today, many application areas require continuous processing of data streams in an efficient manner and real-time fashion. Processing these continuous flows of data, integrating dynamic data with other data sources, and providing the required semantics lead to real challenges. Thus, Linked Stream Data (LSD) has been proposed which combines two concepts: Linked Open Data and Data Stream Processing (DSP). Recently, several LSD engines have been developed, including C-SPARQL and CQELS, which are based on SPARQL extensions for continuous query processing. However, this SPARQL-centric view makes it difficult to express complex processing pipelines. In this paper, we propose a LSD engine based on a more general stream processing approach. Instead of a variant of SPARQL, our engine provides a dataflow specification language called PIPEFLOW which is compiled into native code. PIPEFLOW supports native stream processing operators (e.g., window, aggregates, and joins), complex event processing as well as RDF data transformation operators such as tuplifier and triplifier to efficiently support LSD queries and provide a higher degree of expressiveness. We discuss the main concepts addressing the challenges of LSD processing and describe the usage of these concepts for processing queries from LSBench and SRBench. We show the effectiveness of our system in terms of query execution times through a comparison with existing systems as well as through a detailed performance analysis of our system implementation.

1 Introduction

The availability of modern Sensor Web technologies causes an incredible increasing of published data on the Web. This unbounded flood of data is continuously generated by sensor networks of many spatially distributed devices to monitor variables such as temperature, humidity, pressure, motion, etc., at different locations. Data are also generated by other sources of information such as RSS feeds, micro-blogs, and social networks as well as monitoring applications like system monitoring, network monitoring, and traffic monitoring. Handling all these data in an efficient manner and real-time fashion requires sophisticated methods and tools for continuous query processing.

Processing these continuous flows of streams, integrating such dynamic data with other data sources (i.e., Linked Data), and enriching them with semantic

descriptions for taking ontological knowledge into account, pose several challenges. These challenges have motivated the development of the Linked Stream Data (LSD) idea.

While Linked Data introduces the *Web of Data* concept, LSD adds the idea of publishing triples as *streams of URI-based tuples* or *streams of RDF data*. On the one hand, Linked Data deals with publishing data on the Web as a global database. Then, users can query and interlink multiple sources. On the other hand, Linked Stream Data enables streaming of Linked Data. This concept combines the benefits of Linked data (i.e., adding semantics to data) and the nature of streaming data as dynamic information. Therefore, to give semantics to published tuples of data, the LSD approach supposes to publish triples as *SPO tuples* (i.e., subject, predicate, and object). We can call these URI-based tuples as “*triples-as-tuples*”. Thus, these tuples can be linked and enriched with knowledge from different sources. In LSD, *triples-as-tuples* are produced and evaluated by continuous queries and on-fly processors.

Recently, several LSD engines have been developed, including Streaming SPARQL [9], C-SPARQL [7,6], EP-SPARQL [4] on top of ETALIS, SPARQL_{stream}, and CQELS [16,15]. These engines extend SPARQL to formulate continuous queries over both LSD and Linked Data. However, this approach is limited to the SPARQL style of querying. Furthermore, it lacks a standard SPARQL language for streaming data, i.e., each engine has its own proprietary language extension. Thus, LSD engines cannot currently support a wide range of LSD scenarios.

In this paper, we address the challenges of continuous query processing in the LSD domain by proposing a dataflow engine called PIPEFLOW. In contrast to other systems relying on SPARQL extensions, our engine provides a dataflow specification language which is compiled into native code. Furthermore, our engine uses native stream processing operators, including window-based operators for computing joins and aggregations, filter, projection, etc., to efficiently handle LSD queries. The specifics of LSD are addressed by transformation operators which convert “*triples-as-tuples*” into a “*relational tuples*”, i.e., tuple with a specific schema, and vice versa. In order to integrate tuples from an input stream with static RDF data, a dedicated operator is provided allowing to fetch the RDF data from various endpoints, e.g., local SPARQL processors such as Redland [8] or HTTP-based SPARQL endpoints. Moreover, caching and stream partitioning techniques are exploited to speed-up the execution time of LSD queries. We show the effectiveness of our system in terms of the execution time through a comparison with that of existing systems such as CQELS as well as through a detailed performance analysis of our system implementation using LSBench [17].

The remainder of this paper is structured as follows. We discuss related work in Sect. 2. The requirements of LSD Processing are discussed in Sect. 3. We present our dataflow engine PIPEFLOW and its language in Sect. 4. Some use cases from LSBench and SRBench are explained in order to show our system capabilities in Sect. 5. Finally, we report results of our experimental evaluation in Sect. 6 and conclude our work in Sect. 7.

2 Related Work

Linked Stream Data can be considered as a rich field for research with a lot of challenges to be addressed. In this section, we review the main contributions and related works in LSD area. To clarify the presentation, we present and classify existing systems in three fundamental classes: Stream Processing Engines (SPE) and Complex Event Processing (CEP), Linked Open Data (LOD) and Linked Stream Data (LSD) engines as well as LSD benchmarking.

SPE and CEP: There is an increasing number of applications such as network monitoring, financial applications like banking and trading as well as habitat and environmental monitoring that require on-fly, continuous, and timely processing of data. In order to be able to meet these requirements, numerous Stream Processing Engines and Complex Event Processing Engines have been developed. This includes academic systems such as Aurora [2], Borealis [1], STREAM [5], and S4 [18] as well as commercial systems like IBM Infosphere Streams and Streambase. In SPE's, blocking operators such as joins and aggregation, requiring to have the entire data set available before processing, cause problems with possibly infinite streams. Windowing techniques (e.g., tumbling and sliding windows) can be applied to avoid such blocking by considering only tuples which are valid with respect to some criteria, e.g. a time interval or the number of tuples. Alternatively, CEP deals with data as events and aims to detect and report pattern of events in (temporal) streams such as sequences of specific event types within a given time interval. Systems such as Cayuga [11], SASE [23], SASE+ [3], and Siddhi [21] or commercial systems like Esper [12] provide dedicated event languages to define composite event patterns (higher level information) from primitive events and exploit Non-deterministic Finite Automata (NFA) to detect these event patterns.

LOD and LSD: Several systems have been developed in the context of the Semantic Web for processing Linked Data, for instance, Sesame [19], Jena [13], Redland [8] and YARS [14]. These systems provide a support for manipulating and storing RDF models, parsing RDF/XML, and querying based on a declarative query language. Moreover, a number of LSD engines have recently been proposed. One of these engines called C-SPARQL [7,6] deals with LSD queries by translating them into a static and a dynamic part. It uses a SPARQL plug-in to connect to a SPARQL engine. This plug-in is used to evaluate the static part of the query and to extract the static data only once while the dynamic query is registered in a DSMS to be handled continuously. The current version of C-SPARQL is based on Jena and Esper. EP-SPARQL [4] converts LSD queries into logic programs based on event-driven backward chaining (EDBC) rules. It extends the standard SPARQL for supporting event processing and stream reasoning capabilities by using a SPARQL-like language (i.e., to identify complex events). As a result, EP-SPARQL helps in the detection of more complex events (e.g., temporal sequence occurrence SEQ) based on event streams as well as on static data derived from a background knowledge. Others system such as Streaming SPARQL [9] and CQELS [16,15] extend the standard SPARQL in or-

der to handle RDF streams in LSD queries. The logical SPARQL algebra known in the standard SPARQL 1.1 is extended by these systems in order to deal with streams taking the temporal properties of stream processing into account.

LSD Benchmarking: For evaluating and comparing LSD engines, a few benchmarks have already been proposed: SRBench [24], LSbench [17] and CSRBench [10]. SRBench defines a suite of test queries and metrics to evaluate the performance and properties of the tested LSD engines (e.g., CQELS, C-SPARQL and SPARQL_{stream}). It consists of 17 queries for querying and deriving information from weather stations. These queries have different characteristics, i.e. queries over only stream data sources and both over stream data and static data sources, to ensure that several features of the target systems are checked. In this way, the differences between these systems in terms of supported features can be shown. Sensor measurements (i.e., LinkedObservationData at Kno.e.sis) are considered as stream data whereas the background knowledge (e.g., GeoNames, Linked-SensorData and DBpedia datasets) are considered as static data. LSbench was introduced to evaluate the CQELS engine published in 2012 by comparing its results of some queries (12 queries) with other engines. It proposes three tests to evaluate the LSD engines: functional test, correctness test and maximum input throughput. Therefore, it provides more criteria for testing than SRBench which provides only functional tests. LSbench uses a synthetically generated social graph and five synthetically generated social network streams: GPS position of the social media users, posts and comments, a stream of likes per post, uploaded images, and a stream of likes per photo. CSRBench is an extension of SRBench to address the correctness verification. It was developed to solve the limitations in SRBench by defining new 7 parametrized queries. The authors used a CSRBench oracle engine to generate and compare results of LSD engines and check their correctness.

3 Requirements of Linked Stream Data Processing

Basically, LSD inherits requirements both from data stream processing (DSP) and Linked Data processing. DSP requires to deal with possibly infinite streams of data, e.g., by using windows for determining the validity of data items together with blocking operators such as aggregates and joins. Furthermore, timestamps – either implicit as arrival time or explicit as attribute of data items – as well as timestamp-based operations play an important role for determining the validity of data. Typically, DSP relies on tuple-based data models and a declarative query interface either in the form of a SQL-like query languages or in the form of data flow specifications. Recently, several new distributed stream computing platforms have been developed, aiming at providing scalable and fault-tolerant operation in cluster environments, but (still) lacking a declarative interfaces and, therefore, require to write programs instead of formulating queries.

The core of Linked Data processing is to deal with RDF data, efficient joins across multiple sources, and data transformation tasks. Due to the simple but very flexible triple format, many joins are often required to (re-)construct

(nested) tuples. SPARQL is widely accepted as the standard query language for RDF and Linked Data providing the necessary features for formulating graph patterns but lacks advanced data transformation operations. Based on these observations, the following main requirements can be identified:

- LSD requires to handle weakly and possibly nested structured data.
- Beside the standard query operators, a LSD language has to support window-based operators as well as basic graph patterns (BGP) both for streaming data and static data.
- In order to achieve high throughput, the number of tuples to be processed in the LSD engine should be reduced as much as possible by forming tuples (or graphs) from the triples.
- A LSD engine and language should be extensible by new (user-defined) operators.

In the following, we introduce our approach to address these requirements.

4 A Dataflow Language for LSD

Instead of providing a traditional query language as an extension of SPARQL, we propose to use a dataflow language for LSD processing. Though, this may result in slightly more verbose query formulation, a dataflow language makes it easier to formulate transformation processes and add extensions while still allowing to apply typical query rewriting and optimization techniques.

4.1 Basic Language Concepts

Our PIPEFLOW language is inspired by similar dataflow languages for Hadoop like Pig or Jaql but is not limited to MapReduce programs. In PIPEFLOW, a query is represented by a directed acyclic graph of operators connected by pipes implementing a publish-subscribe mechanism. Each operator is declared by a single PIPEFLOW statement of the following basic form:

```
$out := op($in1, $in2, ...) using (parameters);
```

with the following components:

- an *output pipe* to which the operator publishes its result stream. Each operator can have at most one output pipe and referenced by a variable, e.g., `$out`.
- the *operator* implementing the processing node of the tuples that come through the input pipes. PIPEFLOW provides a large set of predefined operators such as source operators (file and database readers, network socket readers, ...), filters and projections, streaming and relation joins, grouping and

aggregations, sliding and tumbling windows, as well as complex event processing operators. Furthermore, it supports (externally implemented) user-defined operators as well as macros (composite operators). For network communication, we use the 0MQ middleware¹ by providing dedicated operators such as `zmq_publisher` and `zmq_subscriber`.

- multiple input pipes from which the operator receives its input tuples. These pipes are also referenced by a variable, e.g., `$in1`, etc.

Operators are logically connected by sharing a pipe variable, i.e., the output pipe of one operator is used as the input of the next one. Operator-specific clauses can be used to further configure an operator. For instance, the `using` clause is used to pass operator-specific parameters, e.g., the '`filename`' parameter for the file reader operator to specify the input file. The `with` clause allows to specify the schema associated with the output pipe of the operator which is required for source operators such as file and database readers. Furthermore, the `by` clause is used for specifying a predicate which has to be satisfied by each output tuple as for filter, grouping, or joins. In order to specify how an output tuple of the operator is constructed, e.g., in projection and aggregation, the `generate` clause is used.

The following PIPEFLOW script shows an example (Query 10 in LSBench) for counting the number of posts having a certain tag in the last 15 seconds. In this example, the data is read via a file reader operator called `file_source`. The output tuples of this operator are filtered by a predicate (in this case, the predicate component of the triple should be '`sib:hashtag`'). Since we want to compute the aggregation over the last 15 seconds, the output stream is fed into a window with a size of 15 seconds. Then, the tuples are aggregated on their `obj` component to compute the number of posts which have a certain tag. After projecting the components: hash tag and number of posts, the result stream is sent to a 0MQ sink via `zmq_publisher`:

```
$in:= file_source() with (subj string, pred string, obj string)
    using (filename = "rdfPostStream.csv");
$f := filter($in) by pred == "sib:hashtag";
$w := window($f) using (slide_len = 15);
$a := aggregate($w) on obj generate obj, count(subj) as count
    using (slide_len = 15);
$res := zmq_publisher($a)
    using (endpoint = "tcp://*:1235");
```

In addition to ordinary data tuples, PIPEFLOW operators also support punctuations [22] for marking specific states in a data stream, e.g., end of sub-stream, end of window, etc.

The PIPEFLOW engine is implemented as a library of C++ modules. Thus, a PIPEFLOW script is compiled into native C++ code implementing both the

¹ <http://zeromq.org>

dataflow graph (i.e. the execution plan) as well as all tuple type-specific predicates, functions, etc. Note, that a PIPEFLOW script can be compiled to multiple executables which can be deployed to different machines for distributed query processing which we describe in the following sub-section.

4.2 Distributed Processing and Stream Partitioning

Distributed processing in PIPEFLOW is supported by additional higher-level operators such as `exchange` for sending tuples across a network stream (using 0MQ) to another machine or even multiple machines as well as the `map` operator [20]. The `map` operator aims at scalable processing of continuous LSD queries with expensive operations over high-volume streams by parallelizing the execution across multiple nodes. It supports stream partitioning for exploiting data parallelism by splitting the input streams into sub-streams to be processed continuously and independently by replicated queries (i.e., sub-graph of the dataflow program) instances. The `map` operator takes a sub-graph (specified using a macro) as parameter, splits the input stream according to a given key, and executes the sub-graph instances on the sub-streams.

`map` can work in two modes: in the `local` mode, the replicated queries are executed by separate threads on the same machine, whereas in the `distributed` mode the queries are deployed to remote machines by inserting 0MQ send/receive primitives into the dataflow program. Furthermore, the (initial) number of partitions (replicated query instances) can be specified explicitly by the parameter '`num_partitions`'. The following example illustrates the usage of the `map` operator for query 2 in LSBench where the posts posted by a user are notified. In this example, the `map` operator splits the input stream according to the `subj` component of the tuples and executes multiple instances of the sub-query defined in the `do_process` macro in parallel:

```
define do_process ($in) returns $out {
    $f := filter($in) by pred == "sioc:creator_of";
    $res := sparql_join($f) using (endpoint = "file:sibdataset.rdf",
        query = "PREFIX sioc: <http://rdfs.org/sioc/ns#>
                  PREFIX sibu: <http://www.ins.cwi.nl/sib/user/>
                  SELECT * WHERE  :subj sioc:account_of
                    <http://www.ins.cwi.nl/sib/person/p984> .") ;
    $proj:= project($res) generate obj;
};
$in := file_source() using (filename = "rdfPostStream.csv",
    ifs = ",") with (subj string, pred string, obj string);
$res := map($hash) on subj do do_process
    using (mode = "local", npartitions = 10);
```

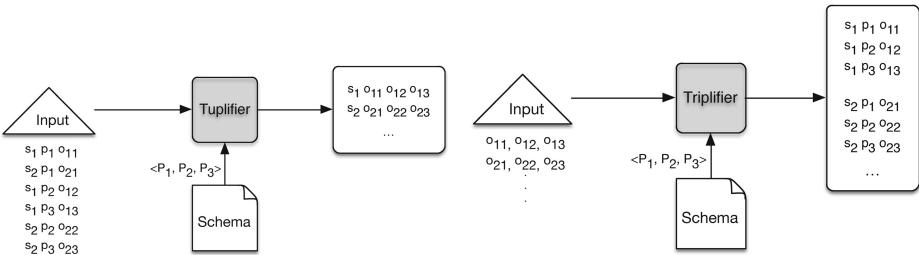
The result produced by the `map` operator is again a single stream which is constructed by merging the results from the sub-query instances. Note, that a distributed version of the query can be easily deployed by simply change the mode to `distributed`. In this case, the PIPEFLOW compiler takes care of generating code for multiple executables including all the necessary send/receive primitives.

4.3 LSD-Specific Operators

Most CEP and SPE systems as well as our system assume the following data stream model. A data stream S is a potentially infinite sequence of elements $S = \langle I_1, I_2, \dots, I_k, \dots \rangle$ where each element $I_j = (e, t)$ is a pair of a tuple e of a given type \mathcal{T} (the schema of the stream) and $t \in TS$ as the timestamp from a discrete ordered time domain describing when the element was observed. Therefore, the timestamp in this sequence is monotonically non-decreasing. As usual, the type \mathcal{T} defines the set of attributes for which each tuple provides data values. For modeling Linked Stream Data, we introduce \mathcal{T}_{LSD} as a special version of tuples type \mathcal{T} representing the components of a RDF triple. Therefore, each element in LSD can be represented as the following:

$$I_j = (\langle subj_j, pred_j, obj_j \rangle, t).$$

We denote tuples of type \mathcal{T}_{LSD} as *primitive tuple* or *triples-as-tuples* to distinguish them from *complete* or *relational* tuples. In addition to the generic operators for arbitrarily structured tuples introduced above, PIPEFLOW provides some dedicated operators to handle this kind of data. These operators are used for transforming between complete tuples and primitive tuples (**tuplifier** and **triplifier**) as well as for joining tuples with data from SPARQL endpoints (**sparql-join**).



Tuplifier Operator. The main point in which LSD processing differs from traditional SPE is the structure of the tuples to be processed. While SPE deals with tuples of the known format \mathcal{T} , LSD publishes and handles URI based tuples which have a raw \mathcal{T}_{LSD} format. However, in real applications, we pay more attention to meaningful higher-level information to be extracted from LSD rather than primitive ones. This requires to put individual primitive tuples into context with other tuples of the same stream. Therefore, PIPEFLOW provides a **tuplifier** operator to transform a set of primitive tuples (triples) into a complete tuple by grouping a set of triples based on a common subject using a specific schema (a list of RDF predicates) as shown in Fig 1. The following example illustrates the idea behind the **tuplifier** operator in the context of social network data. Suppose we have a stream of incoming posts contains the following triples:

```

.....
sibpo:po0 a sib:Post .
sibpo:po0 dcterms:title "null" .
sibpo:po0 dc:created "2010-03-20T01:36:41Z"xsd:dateTime .
sibpo:po0 sib:agent "Motorola" .
sibpo:po0 sib:browser "Chrome" .
sibpo:po0 sioc:ip address "117.55.192.14" .
sibpo:po0 sioc:content "raided a number of locations ...." .
sibpo:po0 sib:hashtag "Operation" .

.....

```

Assuming we want to pack them according to the following schema:

```
[a, title, created, agent, hashtag]
```

Therefore, the resulting complete tuple produced by the statement:

```
$out := tuplifier($in) using
    (uri_schema = "a, title, created, agent, hashtag");
```

would be:

```
[sib:Post, "null", 2010-03-20T01:36:41Z"xsd:dateTime, "Motorola",
    "Operation"]
```

Note, that the above triple data represent only a simple (default) case where all triples belonging to the same complete tuple form a strict sequence. In this case, the tuple is produced as soon as a triple with a different subject is received. If the triple stream is not ordered according to the subject, a window-based strategy can be used. Here, the incoming triples are kept in a window either for a given time period or a number of triples. As soon as the first triple of the group with the same subject is outdated, the complete tuple is produced. Furthermore, sometimes a predicate may have multiple values as objects which is handled by nested tuple fields.

Triplifier Operator. The `triplifier` represents the inverse operator of `tuplifier`. It is used to transform a complete tuple back to a set of triples by specifying a list of RDF predicates. This allows to enrich or annotate the tuples of a stream and raise the ability of making the results more machine readable. The general idea of `triplifier` is shown in Fig. 2 and the operator is specified as follows:

```
$out := triplifier($in) using (subject_field = "key",
    uri_schema = "a list of predicates");
```

where `uri_schema` specifies the RDF predicates for the individual triples. The subject component of the triples can be either

- obtained from a field of the complete tuple specified by the parameter `subject_field` or
- a unique identifier is created and all related triples are connected by a blank node.

The output stream of the operator is always a sequence of triples-as-tuples as strings with one triple for each RDF predicate in the `uri_schema` and the schema \mathcal{T}_{LSD} .

SPARQL Join Operator. The `sparql_join` operator is used to combine tuples from the input stream with data from a RDF data store by executing a join. In the current version of our system, both remote SPARQL endpoints accessible by HTTP requests and local SPARQL processors are supported. The latter is implemented using plugins allowing to integrate a wide range of SPARQL processors. For our experiments, we have selected the Redland engine [8]. Redland is a collection of C libraries that provide a support for parsing, querying and storing RDF models. It includes several modules that implement the storage API such as hashes, trees, files, MySQL databases, etc. We have used the tree module since this module is always present and in-memory. Moreover, it is suitable for larger models and provides indexed storage using several balanced trees for fast querying. In general, the operator can be written as the following:

```
$res := sparql_join($in) using (endpoint = "endpoint config",
                                query = "SPARQL query") with (parameters);
```

The endpoint parameter defines:

- the URL of the remote endpoint (i.e., executing HTTP request) that supports SPARQL queries processing or
- a local file in turtle, RDF or n-triples formats (e.g., `file:data.rdf`) for which the in-memory tree implementation of Redland is used as storage and query engine.

The SPARQL query to be executed for each incoming stream tuple is specified by the `query` parameter. In this query string, placeholders are denoted by `:name` where `name` is a field from the input stream tuple. For example, in query `SELECT ?pred ?obj WHERE { :subj ?pred ?obj }`, the placeholder `:subj` is replaced by the value of field `subj` for each tuple from the incoming stream in order to perform the join. Additionally, the result fields to be added to the output stream tuple are specified in the `with` clause. The `mode` parameter specifies whether an inner join (stream tuples without corresponding tuples in the RDF data are dropped) or an outer join (such stream tuples are filled with NULL values) is performed. This feature is necessary in some queries, for instance in query 8 of LSBench to implement minus or negation filter.

Caching and Indexing: Enriching the incoming LSD with useful data from a SPARQL endpoint requires to process SPARQL queries with a high rate. Typically, a query for each incoming LSD tuple is executed, i.e., for a 100,000 tuples/sec. rate of incoming tuples a corresponding query rate has to be supported. Since static RDF data sets are expected to have a much slower update rate, the result of a query over a RDF data set rarely changes during a continuously running LSD query. We exploit this fact by caching the results of SPARQL joins in an internal hash table. If these results are already contained in the cache (cache hit), the query request can be served directly from the cache without contacting the SPARQL endpoint. Otherwise (cache miss), the query has to be sent to the endpoint. Our system supports several caching algorithms or replacement strategies and exploits indexing for accessing the cache. Both the cache size as well as the strategy can be defined by parameters of the `sparql-join` operator.

5 Use Cases for LSD Processing

In the following, we show the usage of PIPEFLOW for LSD processing. For illustrative purposes we have chosen three examples from LSBench [17,15] and SRBench². These benchmarks aim to serve different kinds of queries applied on streams of social network data (LSBench) and environmental data (SRBench) but share some typical query patterns. We discuss these queries and show how they can be formulated in PIPEFLOW.

Joining Stream Data with Static Data (Query 3 in LSBench): As a basic example of LSD processing, we choose a query that involves a static data set. In this query, the posts posted by all friends of a user (i.e., “<http://www.ins.cwi.nl/sib/person/p984>”) should be selected. Therefore, the join between static and streaming data should be performed.

```
$in := file_source() using (filename = "rdfPostStream.csv")
      with (subj string, pred string, obj string);
$f := filter($in) by pred == "sioc:creator_of";
$res := sparql_join($f) using (endpoint = "file:sibdataset.rdf",
      query = "PREFIX sibu: <http://www.ins.cwi.nl/sib/user/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
SELECT ?user WHERE
      ?user foaf:knows :subj .
      ?user sioc:account_of <http://www.ins.cwi.nl/sib/person/p984> ",
      cache_type="adaptive");
$rename := project($res) generate subj as friend, obj as post;
$out := stream_writer($rename) using (stream = "std::cout");
```

The query is executed by reading the triples from a CSV file, extracting all posts, and generating a stream of tuples of type \mathcal{T}_{LSD} . Next, all tuples satisfying the filter predicate (i.e., ‘`sioc:creator_of`’) are selected. In order to retrieve

² <http://www.w3.org/wiki/SRBench>

the friends of the user, a RDF query on static data (i.e., from the RDF data set `'sibdataset.rdf'`) via the Redland plugin of the `sparql_join` operator is executed. Here, the placeholder `:subj` is replaced each time by the value of field `subj` from the incoming stream tuple in order to perform the join. Finally, the posts and friends are projected and sent to the output. In this example, the internal caching mechanism is used to speed-up the join with the static data.

Sliding window (Query 5 in LSBench): In this query, a notification is issued if a user has been tagged in a photo within a particular period of time that a friend of this user has liked the photo. For this purpose, two different sliding windows on the photo stream and the photo-likes stream are defined to capture the most recent data from the streams, rather than the entire past history when performing the join.

```
$in := file_source() using (filename = "rdfPhotoStream.csv")
    with (subj string, pred string, obj string);
$in2 := file_source() using (filename = "rdfPhotoLikeStream.csv")
    with (subj string, pred string, obj string);
$f := filter($in) by pred == "sib:like";
$f2 := filter($in2) by pred == "sib:usertag";
$s2 := sliding_window($f2) using (type = "range", size = 1);
$s := sliding_window($f) using (type = "range", size = 60);
$jres := symmetric_hash_join($s, $s2) on  $s.subj, $s2.obj
    by $s.subj == $s2.obj;
$proj := project($s) generate subj as photo, obj as friend1,
                    subj_ as friend2;
$res := sparql_join($proj) using (endpoint = "file:sibdataset.rdf",
    query = "PREFIX sib: <http://www.ins.cwi.nl/sib/vocabulary/>
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    SELECT * WHERE :friend1 foaf:knows :friend2 ");
```

First, the query is processed by generating the stream of users which are tagged in the photos and the stream of likes for each photo. These streams are then filtered on the given predicates to restrict the triples to the relevant set. The sliding windows keep all the data in the given period of time. In this example, `(type = "range", size = 60)` denotes a 60 seconds window over the photo stream. The output of these windows are fed into a join operator which implements a symmetric hash join operator for joining tuples from two infinite (but window-based) streams. In our case, the join is used to combine the photo stream with the photo-likes stream. Finally, the results are projected and joined with static data from a SPARQL endpoint to get the friends using the '`foaf:knows`' predicate.

Tuplifying RDF Data (Query 1 in SRBench): This query illustrates the application of the `tuplifier` operator. The input stream representing weather observations which is processed continuously to report rainfall observations. The following example shows the formulation of the query:

```

$in := file_source() using (filename = "observations.csv")
      with (subj string, pred string, obj string);
$out := tuplifier($in)
      with (ob string, sen string, tp string, res string)
      using (uri_schema = "om-owl:procedure, a, om-owl:result");
$out2 := tuplifier($in) with (res string, val string, uom string)
      using (uri_schema = "om-owl:floatValue, om-owl:uom");
$filter := filter ($out) by tp == "weather:RainfallObservation";
$res2 := relation_hash_join($filter, $out2) on $filter.res, $out2.res
      by $filter.res == $out2.res;

```

In this query, the `tuplifier` operator is used two times to generate “complete” tuples which are reconstructed based on a common subject value. The first `tuplifier` groups the incoming observations into a tuple with the schema [`om-owl:procedure, a, om-owl:result`]. The second one prepares a tuple according to the schema [`om-owl:floatValue, om-owl:uom`]. Therefore, two tuple streams are generated in parallel. Since we need only the rainfall observations, the first generated stream is filtered by ‘`weather:RainfallObservation`’. The `relation_hash_join` operator joins the two input streams by treating the second input stream as relation, i.e. storing the tuples in a hash table. Tuples from the other stream are hashed and joined with tuples of the first relation, but are not kept in a hash table.

6 Evaluation

In this section, we present results of an experimental evaluation where we compare our PIPEFLOW engine with the CQELS engine using LSBench. We are not going to test other systems since CQELS has already been tested and compared with other existing systems that also offer integrated processing of LSD. In the experiments reported in [16,17], CQELS has shown better results than C-SPARQL and ETALIS. Additionally, we present results of an evaluation of the techniques presented in this paper such as caching and partitioning techniques. The goal of these experiments is to investigate the performance impact of these techniques compared to a normal processing. Next, we describe our experimental setup, and then report and discuss the obtained results.

6.1 Experimental Setup

For our experiments, we used the social network (SN) scenario introduced in LSBench. LSBench provides a data generator called Stream Social network data Generator (S2Gen) to realistically simulate such data of SNs. It generates two types of data: stream data simulate user activities and static data simulate user metadata, including user profiles and social network relationships. S2Gen offers a range of parameters to produce reasonable input data, but for our experimental purpose, we are interested in the size of the static data and input streams. To evaluate the engines by query expressiveness, we used the proposed

12 queries from LSBench which cover different desired features and aspects of stream queries, i.e., from simple filtering queries to nested complex joins and grouping queries. In all experiments, triple data were fed into the CQELS engine using a loop without any significant delay. For our engine, a comparable approach was chosen by using a `file_source` operator that reads data from a file in order to produce a stream of tuples or “*triples-as-tuples*”.

For all the experiments, we used a single machine which comes with an Intel® Xeon® E5-2630 CPU with 24 cores at clock-speed 2.30GHz. Each core of the processor has 15360K L3 cache, 256K L2 caches, 32K L1 data cache, and 32K L1 instruction cache. The machine runs Ubuntu Linux with kernel version 3.2.0-65-generic and has 128GB main memory. The maximum heap size of JVM instances when running CQELS was set to 100GB. We used the average query execution time as the main performance metric. For this purpose, each query was executed five times and the average of all runs was calculated. We have performed the following three types of experiments:

- *exp 1*: We ran the 12 queries against our system and CQELS and measured the average query execution time. Since we address the problem of scalability over Linked Stream Data, we executed these queries over multiple datasets of different sizes. Therefore, we evaluated them with static data and stream data of 1,000, 10,000, 100,000 and 1M users. Tbl. 1 shows the number of tuples in each dataset. Moreover, we divided these queries into two sets since we have a large set of queries. One set contains stream-only queries and the other contains queries which combined static data with stream data. No caching algorithms were applied on the second set in this experiment.
- *exp 2*: In this experiment, we applied various caching algorithms (i.e., LRU, MRU and adaptive) for some queries in the second set (i.e., Q2, Q3, Q5, Q6, Q9). We used different cache sizes as well as multiple datasets of different sizes and measured the average query execution time.
- *exp 3*: We used query 2 and query 3 from the second set and applied a local stream partitioning technique. This technique partitions the input stream into sub-streams which are processed in parallel in separate threads. Therefore, the comparisons are made between non-partitioned and partitioned queries.

6.2 Results and Analysis

Fig. 3 and Fig. 4 show the results of *exp 1* in a logarithmic scale. The x-axis shows the dataset sizes and queries, whereas the y-axis shows the execution time in (*ms*) for each query. In all cases, our engine achieves significantly better performance; particularly with increasing data volume PIPEFLOW scales much better than CQELS. As we can see from these figures, most of the results in CQELS are missing for two reasons, whether we could not get a final result from executing a particular query (i.e., when executing the query over a large dataset which has a size greater than 10,000 users) or a syntax error was found in the query (e.g., Q9 and Q11). In contrast, PIPEFLOW has results even if the data

Table 1. Number of tuples vs. dataset size

size (# of users)/data (tuples)	photo	post	photo-like	post-like	GPS	static
1,000	260,787	64,650	333,944	198,930	2,528,195	102,955
10,000	2,688,953	923,115	3,450,828	2,990,280	17,427,250	1,396,965
100,000	27,318,184	9,343,184	35,004,384	30,414,460	172,877,630	13,969,421
1M	273,863,410	93,619,255	350,800,224	304,354,230	1,689,156,850	139,760,275

Table 2. Execution times of LSBnech queries over 1,000-users dataset in (*ms*)

engine/query #	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
PipeFlow	63	406	887	171	19,445	64,895	22,672	30,621	163,647	87	532	95
CQELS	2,670	5384	3,236	32,350	235,230	176,996	112,545	164,144	X	17,313	X	18,253

volume is increasing. The only exception is query 9 with 1M-users dataset since it requires many requests to perform the join via the RDF engine. However, we can improve the execution time of this query by utilizing a caching algorithm as we will see in the next experiment. Moreover, the results for 1,000-users dataset show that PIPEFLOW outperforms the CQELS engine significantly by orders of magnitude; sometimes it is over 10 times, especially in set # 2, and over 100 times in set # 1, as shown in Tbl. 2 and the figures.

Exp 2 is divided into two parts. In *exp 2-a*, we compare different caching algorithms and cache sizes over Q2, Q3, Q5, Q6, Q9 queries and 1,000-users dataset to evaluate the performance of the caching techniques. In Fig. 5, the execution times of the runs are shown. The results show that the caching mechanism helps to improve the performance of the execution; for all queries the execution time is improved by increasing the caching size until reaching a point where further increasing does not yield better results. This point in Q2, Q3, Q5, and Q9 was 1,000 whereas in Q6 was 10,000. In *exp 2-b*, different dataset sizes and LRU caching were used. Fig. 6 shows the results of this experiment. This figure highlights a notable issue: by increasing the dataset size, more caching slots are needed in order to improve the performance. Tbl. 3 shows the execution times for the queries over 100,000-users dataset in the points where the best results were achieved and compares them with the execution times for the non-caching queries.

In *exp 3*, the `map` operator was used to partition the input stream on the `subject` attribute. In Fig. 7, the execution times are shown. The x-axis shows the number of cores/threads, i.e., the number of partitions; the y-axis shows the execution times for the queries. The results illustrate that local stream partitioning helps to improve the performance of the LSD processing; for Q2 and Q3 queries the execution time of partitioning queries is better than of the non-partitioning queries. As mentioned in paper [20], the best degree parallelism (i.e., the number of partitions for which the lowest execution time could be achieved, that means, further partitioning does not yield better results) for Q2 was achieved on 6 cores with a value of 16,490 *ms* and 37,495 *ms* for Q3 on 10 cores.

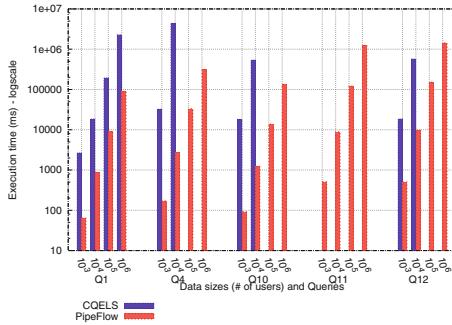
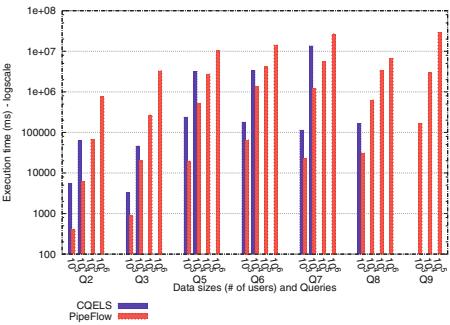
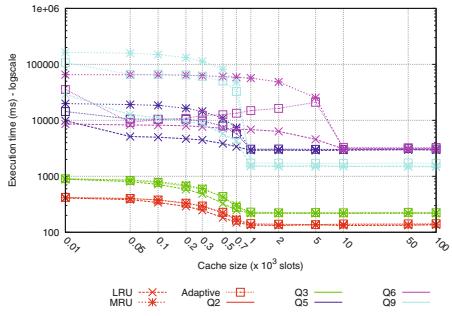
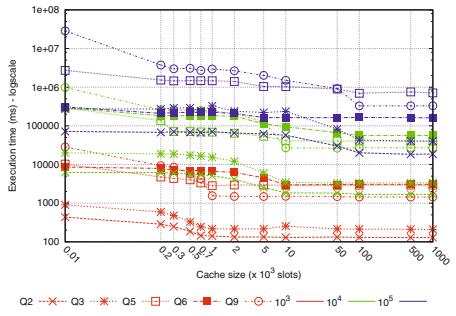
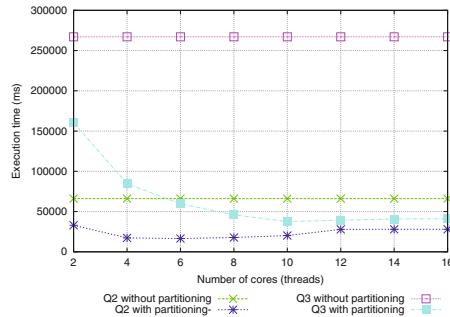
**Fig. 3.** Execution times for query set #1**Fig. 4.** Execution times for query set #2**Fig. 5.** Exp 2-a results over 1,000-users dataset and different caching algorithms**Fig. 6.** Exp 2-b results over multiple datasets and LRU caching**Fig. 7.** Execution times over 100,000-users dataset using local stream partitioning technique

Table 3. Execution times of the queries with and without caching in (*ms*)

caching/query #	Q2	Q3	Q5	Q6	Q9
w/o caching	66,113	267,057	2,722,940	4,123,983	28,645,865
w/ caching	18,666	40,251	711,823	161,564	330,769

7 Conclusion

The nature of Linked Stream Data poses several challenges to query processing. Besides the dynamic behavior of stream data, the specific characteristics of Linked Data have to be taken into account. In this paper, we have presented our PIPEFLOW engine to address these challenges. In contrast to existing LSD approaches relying on SPARQL extensions, PIPEFLOW provides a dataflow language which offers a higher degree of expressiveness particularly useful for data transformation tasks. Our engine supports standard data stream processing operators (e.g., filter, joins, aggregation, window, etc.) as well as specific data transformation operators such as tuplifier and triplifier to efficiently handle LSD queries and enable improved query execution. We have presented the PIPEFLOW language and the specific LSD operators and have illustrated the usage of them to implement queries from the standard LSD benchmarks. We have used LSBench benchmark for an experimental evaluation showing that PIPEFLOW outperforms other systems by orders of magnitude, in some cases over 100 times. In addition, we have investigated the impact of caching and partitioning techniques on the execution times in LSD queries.

Acknowledgment. This research was supported by the German Research Foundation (DFG). We also thank Mazen Salous for developing and implementing the first version of the LSD operators in PIPEFLOW.

References

1. Abadi, D.J., Ahmad, Y., Balazinska, M., et al.: The design of the borealis stream processing engine. In: CIDR 2005, pp. 277–289 (2005)
2. Abadi, D.J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: a new model and architecture for data stream management. The VLDB Journal 12(2), 120–139 (2003)
3. Agrawal, J., Diao, Y., Gyllstrom, D., Immerman, N.: Efficient pattern matching over event streams. In: SIGMOD 2008, New York, NY, USA, pp. 147–160 (2008)
4. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, pp. 635–644. ACM, New York (2011)
5. Arasu, A., Babcock, B., Babu, S., Datar, M.: et al. STREAM: the stanford stream data manager (demonstration description). In: SIGMOD 2003, pp. 665–665. ACM, New York (2003)
6. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: SPARQL for continuous querying. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, pp. 1061–1062. ACM, New York (2009)

7. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Querying RDF streams with C-SPARQL. *SIGMOD Record* 39(1), 20–26 (2010)
8. Beckett, D.: Redland, <http://librdf.org/>
9. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL - extending SPARQL to process data streams. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008. LNCS*, vol. 5021, pp. 448–462. Springer, Heidelberg (2008)
10. Dell’Aglio, D., Calbimonte, J.-P., Balduini, M., Corcho, O., Della Valle, E.: On correctness in RDF stream processor benchmarking. In: Alani, H., et al. (eds.) *ISWC 2013, Part II. LNCS*, vol. 8219, pp. 326–342. Springer, Heidelberg (2013)
11. Demers, A., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., White, W.: Cayuga: a general purpose event monitoring system. In: *CIDR 2007*, pp. 412–422. VLDB (2007)
12. EsperTech. Event stream intelligence: Esper & NEsper, <http://www.esper.codehaus.org/>
13. A. S. Foundation. Apache Jena, <https://jena.apache.org/>.
14. D. E. R. Institute. YARS, <http://sw.deri.org/2004/06/yars/>
15. Le Phuoc, D.: A native and adaptive approach for linked stream data processing. PhD thesis, NUI Galway (March 2013)
16. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) *ISWC 2011, Part I. LNCS*, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
17. Le-Phuoc, D., Dao-Tran, M., Pham, M.-D., Boncz, P., Eiter, T., Fink, M.: Linked stream data processing engines: Facts and figures. In: Cudré-Mauroux, P., et al. (eds.) *ISWC 2012, Part II. LNCS*, vol. 7650, pp. 300–312. Springer, Heidelberg (2012)
18. Neumeyer, L., Robbins, B., Nair, A., Kesari, A.: S4: distributed stream computing platform. In: *ICDMW 2010*, Washington, DC, USA, pp. 170–177 (2010)
19. OpenRDF. Sesame, <http://www.openrdf.org/>
20. Saleh, O., Betz, H., Sattler, K.-U.: Partitioning for scalable complex event processing on data streams. In: Bassiliades, N., Ivanovic, M., Kon-Popovska, M., Manolopoulos, Y., Palpanas, T., Trajcevski, G., Vakali, A. (eds.) *New Trends in Database and Information Systems II. AISC*, vol. 312, pp. 185–197. Springer, Heidelberg (2015)
21. Sriskandarajah, S., Kasun, G., Isuru, L.N., Subash, C., Srinath, P., Vishaka, N.: Siddhi: a second look at complex event processing architectures. In: *GCE 2011*, pp. 43–50. ACM, New York (2011)
22. Tucker, P.A., Maier, D., Sheard, T., Fegaras, L.: Exploiting punctuation semantics in continuous data streams. *IEEE Trans. Knowl. Data Eng.* 15(3), 555–568 (2003)
23. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: *SIGMOD*, pp. 407–418 (2006)
24. Zhang, Y., Duc, P.M., Corcho, O., Calbimonte, J.-P.: SRBench: A streaming RDF/SPARQL benchmark. In: Cudré-Mauroux, P., et al. (eds.) *ISWC 2012, Part I. LNCS*, vol. 7649, pp. 641–657. Springer, Heidelberg (2012)

Applying Semantics to Optimize End-User Services in Telecommunication Networks

Liam Fallon¹, John Keeney¹, and Declan O’Sullivan²

¹ Network Management Lab, Ericsson, Athlone, Co. Westmeath, Ireland

² Knowledge & Data Engineering Group (KDEG), Trinity College, Dublin, Ireland

Abstract. This paper describes Aesop, a semantically-enhanced approach for optimizing telecommunication networks to manage over-the-top end-user services such as web-browsing and video-streaming, according to the experience expectations of individual users. Aesop is built using technologies already available in the semantic ecosystem. In this paper, we explain our semantic approach to this problem domain and describe how we applied semantic technologies in the design and implementation of our system. We give an overview of an implementation and evaluation of Aesop. We describe our experiences in using semantic technologies, and explore the potential and limitations of semantic technologies in the telecommunication management domain.

1 Introduction

In this work we present the AESOP engine, which uses semantic technologies and models in monitoring and managing telecommunications networks and the services that use them. Semantic modelling allows the structure, meaning, and references of models to be captured using ontologies [4], which can then be executed. Operations can be carried out on models directly using semantic frameworks, queries and rules can be run on models, and reasoners can automatically deduce knowledge and relationships from the current set of knowledge in a model. The strength of semantics is that complex relationships across disparate knowledge domains can be modelled, and knowledge can be captured [4].

In the Aesop engine, monitoring operations determine end user experience and context which are then reported periodically into appropriate semantic snapshot buckets. Analysis operations semantically analyse and update how well each reported snapshot complies with expectations for the given context. Optimization planning and execution operations plan optimizations by semantically inferring and querying the compliance of the current running session set to determine if expectations are being met appropriately. If not, actions are executed on the service delivery network to prioritize delivery of high priority sessions.

2 Background and Related Work

Monitoring metrics from telecom network nodes are highly standardized [1], and provide valuable information about the state of any network being used to deliver

over the top (OTT) services (e.g. web-browsing) to users, however these metrics rarely provide information about the quality of the 'service' (QoS) (e.g. packet-loss rate), and never provide information about how the quality of experience (QoE) (e.g. webpage-load time) of the service users. Reporting directly from service terminals is the most accurate way of evaluating end user service context, with application-specific information such as video frame rates, latency, jitter in addition to contextual information such as user identity, location, CPU load, memory usage, and battery levels.

The overall goal of our work is optimization of end-user service delivery, an immature field in practice and research [18]. Approaches have been proposed for optimization of centrally controlled end-user services using centralized session control mechanisms [18], but such approaches do not work when such an authority is not present. Other approaches such as HTTP adaptive streaming optimize single sessions at the expense of all other sessions [15]. A learning controller [12] can optimize across end-user service sessions, and policies can be applied across sets of sessions to improve the overall compliance of the running session set with expectations. Policy-based [19] and Semantic [16] approaches have been described for traffic control of streams in HANs, but these approaches do not take the Quality of Experience (QoE) for service end-users, or their context, into account. Optimization in SON for mobile broadband networks as surveyed in [3] also focuses on optimization of network traffic rather than end-user services.

While automated and autonomic approaches for network and service management are many and varied [2], successful adoptions of semantic technologies are few. As already mentioned some approaches focus on semantic enrichment of management models for richer context modelling and improved interoperability with other models or datasources. Some semantically enhanced policy frameworks [10][7] are often based on or are similar to semantic rule languages like SWRL. Approaches such as the BREIN ontology [14] focus on Service Level Agreements (SLA) compliance. Although this ontology does not model end user service delivery, the approach of using a simple core ontology providing a common understanding of concepts is compelling. Rodrigues et al. [17] describe an ontology for managing network service quality by monitoring SLA compliance but it has no concept of end user services, and uses a direct mapping of QoS parameters to quality of experience (QoE). Flexible SWRL-based mappings of QoS to QoE are used in NetQoSOnt[5] but it does not take into account sharing of resources across service sessions.

3 Aesop: Semantic End User Service Management

The Aesop approach has a *Service Knowledge Base* at its core and runs an autonomic management loop around that core. The knowledge base contains knowledge for the end-user service management domain. Service Expectations are loaded into the knowledge base by the system user. Service Experience and Service Context *Monitored* knowledge describing how well end user services are operating is loaded into the knowledge base using semantic lifting if it is not in a

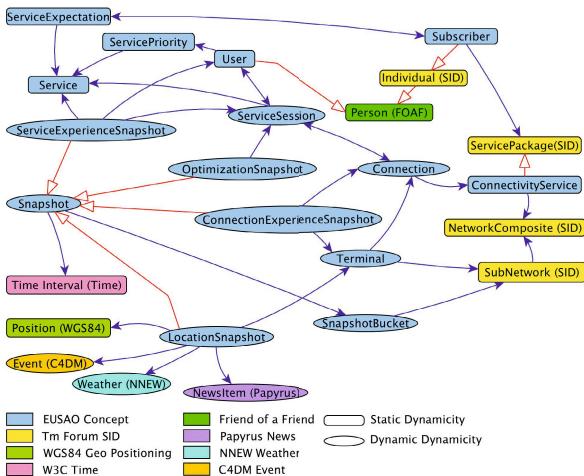


Fig. 1. Core Concepts in the EUSAO ontology

semantic form or directly if it is already in a semantic form such as RDF. Service *Analysis* knowledge is created by Semantic Service Analysis algorithms that check how well the service experience being experienced by end users complies with users' service expectations in the service context in which they are running. Service Optimization *Planning* and *Execution* knowledge is created by Semantic Service Optimization Planning and Execution algorithms that determine what optimization plans should be applied to the network in the form of traffic controls to improve the manner in which the network is delivering end user services. These traffic controls are applied to the network, thus completing an iteration of a autonomic loop in near real time. In this paper, we focus on the semantic aspects of the Aesop approach. For a description of the Aesop approach from a network management point of view, see [9].

3.1 The EUSAO Ontology and the Knowledge Base

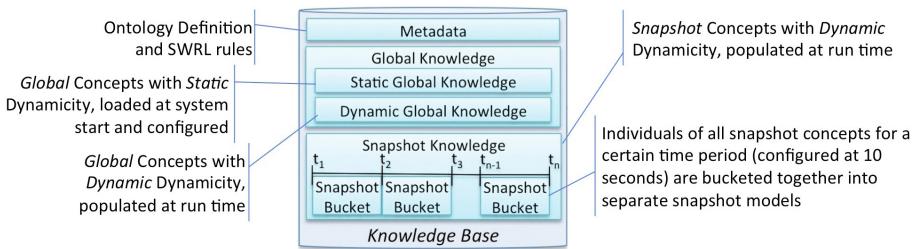
Fig. 1 shows a simplified version of the EUSAO (End User Service Analysis and Optimization) ontology [8][9], models concepts and relations for end user service management. The ontology models two temporal aspects of concepts; their *dynamicity* and their *temporality*. If the individuals of a concept can only be changed by configuration, the concept has *static dynamicity*; a concept has *dynamic dynamicity* if its individuals are changed at run time by a system using the ontology. If individuals of a concept are not created periodically during execution, that concept has *global temporality*; individuals of concepts created periodically have *snapshot temporality*. Examples of concept temporality are shown in Table 1.

The knowledge base is structured into separate models using the temporal aspects of concepts, as shown in Fig.2. Queries and rules that use specific parts

Table 1. Examples of Temporal Aspects of Concepts

Concept	Dynamic	Temporal
Expected Web Browsing Page Load Time	Static	N/A
Web Browsing Session	Dynamic	Global
Browser of a Web Browsing Session	Dynamic	Global
Average Page load Time for a Time Interval	Dynamic	Snapshot

of the knowledge base can be directed to specific sub-graphs of the knowledge base. The concept of a snapshot is fundamental to the performance of the Aesop KB. The KB maintains a graph model for each snapshot bucket (Fig. 2). All individuals with **dynamic** dynamicity and **snapshot** temporality for the time period of a snapshot bucket are stored in that snapshot bucket model, thus reducing the numbers of individuals requiring reasoning, thus reducing execution times. Aesop functions use semantic operations that act on temporary reasoned models created from subsets of the KB, usually composed of metadata, both global knowledge models, and one or more snapshot bucket graphs.

**Fig. 2.** The Aesop Knowledge Base

The efficiency of the Aesop Knowledge Base arises from its use of *snapshot bucket* models. It maintains a model for each snapshot bucket (see Fig. 2). All snapshot individuals for the time period of a snapshot bucket are stored in that snapshot bucket model. These individuals may reference other individuals in their snapshot bucket or global knowledge model individuals, but may not reference individuals in other snapshot buckets. This restriction, along with the separate pre-processing of schema data and instance data with static dynamicity, ensures that there is no requirement to support reasoning across bucket boundaries. Snapshot Bucket models are retained for a certain retention period, set at 10 minutes in this implementation.

There are two alternative methods to implement a bucket approach: using named graphs in a single store, or using separate partitioned stores for each bucket. Here we chose the latter approach, thus lessening the reasoning overhead and allowing some degree of parallelization as one bucket can be safely queried while the next bucket is being prepared. Using this approach greatly reduces the size of models used for semantic operations, thus increasing their performance.

An alternative to this bucket approach is the use of continuous stream-based reasoning [13]. However some issues with such a continuous stream based approach include restricted reasoning expressivity, high resource usage, less predictable performance characteristics, and the need to perform batch-based (uplift) processing of logs. This work shows that careful design, partitioning and flexible bucketing of the knowledge base and instance data removes the need for semantic stream processing in this work. In fact, the bucketing approach used in AESOP is inspired by temporal-tumbling-windows seen in traditional Stream Processing and Complex Event Processing engines.

3.2 The Semantic Algorithms

The autonomic functions of Aesop are designed and realized as semantic algorithms that act on the Knowledge Base. Each algorithm follows a three-phase general form: 1) create a temporary Jena graph model made up of a subset of the models in the knowledge base and run a reasoner on that model to infer knowledge; 2) use semantic queries, rules, and Jena operations to manipulate the knowledge in that temporary model; and 3) store new knowledge back into the knowledge base as asserted individuals. The Pellet reasoner [6] was used in conjunction with Jena for reasoning. The Jena implementation of SPARQL was used for queries and SWRL was used for rules. Aesop was implemented in Java.

The Semantic Monitoring Algorithm. When Aesop detects that one or more reports has been received from client terminals, semantic lifting (as described in [8]) is used where SAWSDL [22] annotations on the XML schema of the terminal reports trigger XSL transformations on the reports to semantically lift portions of the terminal reports into RDF. This RDF is stored in a temporary inference enabled graph, along with necessary metadata and global knowledge models. SWRL rules are then triggered to infer new dynamic knowledge and snapshot knowledge about the services and the network. The Knowledge Base handles creation of snapshot models and storage of individuals in the appropriate snapshot model. At startup, a snapshot interval is defined ($t_n - t_{n-1}$ in Fig.2). The time stamp of an individual is used to decide which snapshot an individual should be stored in. If the time stamp is greater than t_n on the most recent snapshot model, then a new snapshot model is created.

This algorithm is efficient because it restricts the scope of reasoning and rule execution to Metadata and Global Knowledge during insertion of new monitoring knowledge; consideration of existing snapshot knowledge is not required. The knowledge handled by the algorithm can be extended by modifying SAWSDL annotations and XSL transformations used during lifting or by amending SWRL rules triggered when knowledge is loaded into the temporary model.

The Semantic Analysis Algorithm. Firstly, a temporary reasoned model is created composed of the metadata, global knowledge, and the relevant snapshot. A generic SPARQL query is run on the temporary model to find new or updated service experience snapshots and their equivalent service expectations.

Then, the service type of each service experience snapshot is used to look up a service specific query for each service experience snapshot found. This query is run on the temporary model to determine the compliance of that particular service experience snapshot individual with that particular service expectation individual. The compliance of that service experience snapshot individual is then updated in the Knowledge Base.

The algorithm is efficient because reasoning and querying is executed on the knowledge of a single snapshot thus reducing the number of individuals over which semantic operations execute. The algorithm performs a single Jena read operation and a single Jena write operation for each snapshot bucket. The algorithm can be adjusted at run time because the SPARQL queries that perform its calculations can be changed at any time.

The Semantic Optimization Planning and Execution Algorithm. runs periodically to determine if and what optimizations should be performed on the network to optimize the currently running set of end user services. Firstly, the algorithm builds a temporary model using metadata, global knowledge, and snapshots covering the period of time into the past that the optimization algorithm should consider (*lookback interval*). Next, if there has not been a recent application of an optimization action, a SPARQL query checks if an optimization action is merited by determining if high priority sessions are being degraded by low priority sessions. If so, a list of sessions for throttling is selected. Optimization execution applies or releases throttling actions on the network for sessions selected for throttling by invoking scripts that send commands to the network. The status of the sessions on which throttling has been executed is updated in the knowledge base.

The algorithm is efficient because temporary models use snapshot models that cover the lookback interval to avoid the need to reason and query over the entire snapshot knowledge in the knowledge base. Tests for stability and recent optimization are implemented as SPARQL queries that can be easily added, removed or modified.

4 Results and Experiences

In a set of experiments in a heavily loaded Home Area Network, documented in [9], it is shown that Aesop is effective at improving how well higher-priority services comply with user expectations while lower priority services are automatically throttled. It was also observed that the execution time of operations is related to the size of the temporary models being used and the complexity of the semantic algorithm. The most straightforward Analysis algorithm, acting as it does on a single snapshot model, executes more rapidly than the Monitoring algorithm, which must parse, lift, and place knowledge in the appropriate global and snapshot models. The most complex Optimization Planning and Execution algorithm takes the longest time to execute.

Evaluation shows that Aesop can run semantic operations that manage end user services in a home area network in near real time using readily available

and unmodified semantic frameworks, libraries, and toolsets. Average monitoring, analysis, and optimization times of the order of two to four seconds were observed with CPU usage at about 20% when four service sessions were running. During execution it was observed that Aesop used approximately 4GB of RAM in passive runs, and 5.5GB in active runs. Additional experiments also show that AESOP can run adequately with lower RAM, but requires more aggressive memory management, thus affecting execution times to some extent.

In general, our experiences in using a semantic approach and semantic technologies in Aesop have been positive. We have found that the EUSAO ontology allowed us to capture the relationships and interactions of the problem domain as well as its structure in a manner that is not possible using UML or models in programming languages. Taking a semantic approach allowed us to design Aesop so that all knowledge handling is configurable, either by changing relations or constraints in the EUSAO ontology, or by changing rules or SPARQL queries.

All queries in the prototype implementation of Aesop were implemented in SPARQL using Jena ARQ. The language was found to be flexible, expressive and powerful. Rather complex queries were relatively easy to specify in the language. The performance of queries was dependent on whether they were run on unreasoned or reasoned models. In an informal test it was found querying using a SPARQL query or directly using the Jena API had very similar performances for sensible queries.

Rule specification and application was also straightforward, particularly during knowledge lifting and loading. In the Aesop prototype SWRL rules were executed in the Pellet reasoner whereby SWRL rules in a reasoned model are fired once a pertinent individual is loaded. This approach for rule implementation is certainly convenient and useful.

The Jena model API was used extensively in the prototype implementation of Aesop. The model and querying APIs were intuitive and well documented. The model API was rich enough to access and store knowledge in models at various levels of detail. The performance of storage and access operations to and from Jena unreasoned models was not the main bottleneck in the system, and was adequate for the prototype implementation.

The ability to "execute" the model using reasoning at run time is the greatest strength of taking a semantic approach. Reasoning is resource intensive, but the knowledge base structure used in Aesop allowed us to apply reasoning in our semantic algorithms in an efficient manner. As expected, the performance of semantic operations on models being reasoned over by the Pellet reasoner (via Jena) were slower than those executed towards unreasoned models, of the order of hundreds of milliseconds. The performance of the reasoner was adequate for the prototype implementation because the Aesop design approach ensured the size of reasoned models was small and because reasoning load was reduced due to the decision not to load the metadata for referenced models at run time. This further illustrates the benefits of employing different reasoning strategies for metadata, schema and instance data. Since most of the time budget for a semantic operation is attributable to operations towards reasoned models,

tuning of the Pellet reasoner or selection of a more targeted reasoner such as that discussed by Tai et al. [20] to improve query time would significantly improve the performance of the Aesop prototype. Despite this the time, CPU and memory results show that the Aesop approach is an effective and practical way of applying semantics in network management systems in near real-time.

In our experience, the most complex and difficult semantic task undertaken by us was design and implementation of the EUSAO ontology. Developing a compete working ontology is a complex task and requires a significant degree of experience in using semantic modelling technologies such as RDF and OWL. Although Protégé is a powerful modelling tool, it is not as polished as modelling tools commonly used for UML modelling. One particularly useful feature of Protégé was the ability to use the Pellet Reasoner at design time to validate the EUSAO ontology during implementation and to validate global static individual data. However, it was much easier to input large amounts of individual data by manually specifying individuals directly in RDF/XML, then validating and visualising using Protégé and Pellet. However, once the ontology was specified, applying reasoning, SWRL rules, and SPARQL queries was straightforward and such tasks could, in our opinion, be assigned to relatively novice developers.

5 Summary and Future Work

The Aesop approach applies semantic technologies in an autonomic system for management of end user services in telecommunication networks. The design of Aesop shows that a semantic approach can be taken in the design of management systems and has allayed concerns in the network management community with using semantic techniques [21][11]. The Aesop approach provides a reference architecture for other autonomic management systems that wish to take a semantic approach and provides a benchmark for assessing such work. The structure of the ontology is core to the efficiency of Aesop because it allows the knowledge base to be partitioned in a manner that facilitates efficient access. Once the correct ontology structure was established, specification of powerful queries and rules on the ontology became relatively straightforward.

In future work, we will examine how we might modify the EUSAO ontology to increase reasoning performance. In the telecommunications network management domain, incoming information is generally produced by network elements such as switches and base stations and is trusted and well structured. Therefore, ontologies in the network management domain can be designed with minimal constraint checking. Another avenue of future work to be pursued is an analysis of what frameworks and reasoners fit best in the network management domain and how best to tune such semantic frameworks and reasoners.

Acknowledgments. This work was partially supported by Science Foundation Ireland under the FAME SRC Grant No 08/SRC/I1403.

References

1. 3GPP: Telecommunication Management; Performance Management (PM); Performance Measurements - GSM. Tech. Rep. TS 52.404 (March 2011)
2. Agoulmine, N.: Autonomic Network Management Principles: from Concepts to Applications. Elsevier Academic Press (2010)
3. Aliu, O.G., et al.: A Survey of Self Organisation in Future Cellular Networks. *IEEE Communications Surveys Tutorials* 15(1), 336–361 (2013)
4. Allemand, D., Hendler, J.: Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann Publishers Inc. (2008)
5. Cé Júnior, J., et al.: A Semantic Approach for QoS Specification of Communication Services using QoE Parameters. *Journal of the Brazilian Computer Society*, 1–15 (November 2012)
6. Clark&Parsia: The Pellet Reasoner (2013)
7. Davy, S., et al.: Policy Interactions and Management of Traffic Engineering Services Based on Ontologies. In: LANOMS, pp. 95–105 (September 2007)
8. Fallon, L., O'Sullivan, D.: Using a Semantic Knowledge Base for Communication Service Quality Management in Home Area Networks. In: NOMS (April 2012)
9. Fallon, L., O'Sullivan, D.: The Aesop Approach for Semantic-based End-User Service Optimization. *IEEE TNSM* 11(2) (June 2014)
10. Jennings, B., et al.: Towards Autonomic Management of Communications Networks. *IEEE Communications Magazine* 45(10), 112–121 (2007)
11. Keeney, J., et al.: Approaches to Relating and Integrating Semantic Data from Heterogeneous Sources. In: WI-IAT, vol. 1, pp. 170–177 (August 2011)
12. Latré, S., et al.: An Autonomic Architecture for Optimizing QoE in Multimedia Access Networks. *Computer Networks* 53(10), 1587–1602 (2009)
13. Margara, A., et al.: Streaming the web: Reasoning over dynamic data. *Web Semantics Journal* 25 (2014)
14. Muñoz Frutos, H., Kotsopoulos, I., Vaquero Gonzalez, L.M., Rodero Merino, L.: Enhancing service selection by semantic QoS. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 565–577. Springer, Heidelberg (2009)
15. Oyman, O., Singh, S.: Quality of experience for http adaptive streaming services. *IEEE Communications Magazine* 50(4), 20–27 (2012)
16. Rana, A., Jennings, B.: Semantic uplift of monitoring data to select policies to manage home area networks. In: AINA, pp. 368–375 (March 2012)
17. Rodrigues, C., et al.: An Ontology for Managing Network Services Quality. *Expert Systems with Applications* 39(9), 7938–7946 (2012)
18. Sterle, J., et al.: Application-based ngn qoe controller. *IEEE Communications Magazine* 49(1), 92–101 (2011)
19. Sventek, J., et al.: An information plane architecture supporting home network management. In: Integrated Network Management (IM). IEEE (May 2011)
20. Tai, W., Keeney, J., O'Sullivan, D.: COROR: A COmposable Rule-Entailment Owl Reasoner for Resource-Constrained Devices. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 212–226. Springer, Heidelberg (2011)
21. de Vergara, L., et al.: Ontology-Based Network Management: Study Cases and Lessons Learned. *Journal of Network and Systems Management* 17, 234–254 (2009)
22. W3C: Semantic Annotations for WSDL and XML Schema. Tech. rep. (August 2007)

An Ontology-Based Data Exploration Tool for Key Performance Indicators

Claudia Diamantini, Domenico Potena, Emanuele Storti, and Haotian Zhang

Dipartimento di Ingegneria dell'Informazione

Università Politecnica delle Marche

via Brecce Bianche, 60131 Ancona, Italy

{c.diamantini,d.potena,e.storti}@univpm.it, zhanghaotian@gmail.com

Abstract. This paper describes the main functionalities of an ontology-based data explorer for Key Performance Indicators (KPI), aimed to support users in the extraction of KPI values from a shared repository. Data produced by partners of a Virtual Enterprise are semantically annotated through a domain ontology in which KPIs are described together with their mathematical formulas. Based on this model and on reasoning capabilities, the tool provides functionalities for dynamic aggregation of data and computation of KPI values through the formula. In this way, besides the usual drill-down, a novel mode of data exploration is enabled, based on the expansion of a KPI into its components.

1 Introduction

In recent years the capability to innovate and collaborate among different enterprises is more and more considered a crucial skill to react more dynamically to market changes and reduce risks [1]. However, one of the main issues to overcome, especially in the management of temporary networks of enterprises (i.e., Virtual Enterprises, or VE), is the integration of heterogeneous data, and the need to evaluate common Key Performance Indicators (KPI) that are capable to measure performances of the whole VE. In such distributed scenarios, besides heterogeneities due to the use of different terminologies, procedures and processes, each enterprise usually adopts a different set of KPIs to measure their own performances. This introduces a new type of heterogeneity hindering the possibility to monitor comparative performances for the whole VE.

This work is conceived within the European project BIVEE¹, that is targeted at supporting Virtual Enterprises in the achievement of common innovation projects. With the aim to address the above-mentioned issues, the development of the BIVEE platform follows an ontology-based approach for data access, that is frequently used to tackle the problem of integrating and accessing heterogeneous sources, abstracting from how data are maintained. In such a way, a data layer is responsible of storing actual KPI data produced by enterprises, while a

¹ <http://bivee.eu>

conceptual model is shared among all the partners and provides a common terminology used to express user requests. Finally, the mapping between the two layers allows to rewrite ontological queries in a language suitable for the data layer.

The conceptual model is here represented by KPIOnto, an ontology devoted to formalise the domain of KPIs and their relations in a logic language. Differing from most of available proposals in the Literature (e.g., [2–5]), in BIVEE KPIs are also formally described through the mathematical formulas needed for their calculation starting from other KPIs. In such a way, different heterogeneities among enterprises can be solved by semantic enrichment of data, or by reasoning over the formulas.

On the top of BIVEE platform, in this work we introduce KPI Explorer, a tool designed to provide users with access to KPI data in an intuitive and informative fashion. Through the reference to KPIOnto, the tool supports users in the graphical composition of a query. Unlike traditional reporting softwares or data exploration tools, the back-end functionalities of KPI Explorer, by exploiting the ontology, are aimed at (1) query rewriting and its execution on the Data Storage, but also to (2) dynamically, transparently and automatically calculate KPI values that are not materialized, but that can be inferred from those available, providing more complete results to users; such advanced functions are enabled by reasoning capabilities that exploit ontological relations among concepts as well as manipulation of mathematical KPI formulas. As for the former, for instance, let us suppose that a KPI is measured by an enterprise on a monthly basis. Given that the concept of “quarter” is described in KPIOnto and every quarter is linked to its corresponding months, it is possible to evaluate the KPI by aggregation of these ones. As regards the latter, let us consider two enterprises willing to compare performance about a certain KPI that, however, is provided just by one of the two; given that formulas are explicitly represented, if the KPI can be obtained for the other enterprise through some formula starting from the KPIs that are currently provided, it will be possible to dynamically compute the final value for both of them. Finally, users can refine the results by applying both classical operators like roll-up/drill-down, and a novel “indicator drill-down” operator, that allows to expand an indicator into its components, allowing the user to gain a better understanding of KPI trends.

This work is structured as follows: Section 2 discusses the most relevant contributions in the Literature, while in Section 3 we introduce those modules of the BIVEE architecture that are central to this work, namely the ontology KPI-Onto and the Raw Data Handler. Section 4 introduces the Explorer, discussing both back-end functionalities for query rewriting and reasoning aimed at the evaluation of KPIs and at the manipulation of their formulas. The front-end of the tool is also described and some examples are given. Section 5 presents some experimentations carried on to evaluate the efficacy and efficiency of the proposed approach. Finally, Section 6 provides some final remarks.

2 Related Work

The sharing of data coming from distributed, autonomous and heterogeneous sources arises problems of data inconsistencies and asks for methods to integrate them in a unified view. Given that local schemas are independently developed, they usually have different structure and terminology and several syntactic, structural, and semantic conflicts can occur during schema integration [6].

Several approaches have been proposed in the Literature for schema matching, that is the basic problem in database integration. While real data are produced by the sources, a common approach is to rely on a global schema to obtain an integrated and virtual view, through global-as-view (GAV) and local-as-view (LAV) approaches [7]. Recently, due to the explosive growth of distributed applications, the need of semantic integration is becoming more and more critical. In fact, semantic heterogeneity may persist even if both syntactic and schemas heterogeneities do not occur (e.g., naming the same concept differently). Semantic data integration relies on conceptual and machine-understandable representation of data and their relationships to avoid heterogeneities and allow interoperability. The main methodologies used in the Literature refer to a single ontology approach in which all source schemas are mapped to a central ontology, or a multiple-ontologies approach, with mapping among the local schemas and no shared global view, closely related to GAV and LAV approaches [8]. Ontologies are not only useful as a global conceptualization, but also to support the definition of semantic mappings between the local sources and the global ontology, and to support the user in the formulation of high-level queries over the global schema and its rewriting into local queries. Integration is far more complex when the data models of the sources are multidimensional, as in the case of the present work and in the data warehouse field. Multidimensional models are specifically suited to support data analysis rather than perform on-line transactions, and categorize data as facts with associated quantitative measures, or as a hierarchy of dimensions that describe the facts.

The multidimensional model takes into account the aggregative aspect, defining a data cube as a multi-level, multidimensional database with aggregate data at multiple granularities [9]. The definition of powerful OLAP operators like drill-down directly comes from this model. Semantic representations of the multidimensional model have been recently proposed [9–12] mainly with the aim to reduce the gap between the high-level business view of indicators and the technical view of data cubes, to simplify and to automatize the main steps in design and analysis. In particular, support to data cube design is considered in [13], while improvement of OLAP functionalities are presented in [4].

Although the complex nature of indicators is well-known, the compound nature of indicators is far less explored. Proposals in [2, 13–18] include in the representations of indicator properties some notion of formula in order to support the automatic generation of customized data marts, the calculation of indicators [13, 14, 16], or the interoperability of heterogeneous and autonomous data warehouses [2]. In the above proposals, formula representation does not rely on logic-based languages, hence reasoning is limited to formula evaluation by

ad-hoc modules. No inference mechanism and formula manipulation is enabled. Formal, logic-based representations of dependencies among indicators are proposed in [17, 18]. These properties are represented by logical predicates (e.g. *isCalculated* [18], *correlated* [17]) and reasoning allows to infer implicit dependencies among indicators useful for organization modeling, design, as well as reuse, exchange and alignment of business knowledge. An ontological representation of indicator formulas is proposed in [15] in order to exchange business calculation definitions and to infer their availability on a given data mart through semantic reasoning, with strong similarities with our previous work [19].

3 Knowledge-Centric BIVEE Platform

The BIVEE project relies on a knowledge-centric approach in which the semantic layer, namely the PIKR [20], represents the foundation supporting advanced functions of the whole environment. It maintains semantic metadata for all the entities of the VE (from actors to products, from innovative ideas to production plans) and offers advanced services. In this Section we briefly discuss the back-end modules of BIVEE infrastructures that are more relevant in the context of this paper. They include KPIOnto, the ontology that is used to represent KPIs and dimensions in the platform, and the Raw Data Handler, where data from enterprises are semantically enriched and stored.

3.1 KPI Ontology

A number of Performance Indicators definitions have been introduced, including glossaries provided by researchers and research groups [21, 22], and international and national public bodies (e.g. OECD²). While these cannot be regarded as proper models, due to their informal nature, also some reference frameworks for supply chain management were proposed, e.g. the Supply Chain Operations Reference model (SCOR) [23] and Value Reference Model (VRM)³ that are the two most comprehensive and widely adopted (see also [24] for a more detailed list of performance measurement approaches and KPI description).

Therefore, in order to derive the main properties for the development of KPI-Onto [25], an ontology devoted to formally describe indicators, we referred to VRM with some additional contributions from other glossaries, and to the multidimensional model of data warehouse domain. This choice is justified by three main reasons: (1) the VRM explicitly addresses networked enterprises, that is the reference scenario of this work, (2) the model provides the most complete and detailed description of the properties of indicators, and (3) the BIVEE project itself mainly referred to the VRM model to develop its business innovation reference framework [24].

² <http://www.oecd.org/std/leading-indicators/glossaryforoecdcompositeleadingindicators.htm>

³ <http://www.value-chain.org/en/cms/1960>

Following the VRM model, in KPIOnto KPIs are arranged in a taxonomy according to the enterprise area of intervention or process (e.g. customer, corporate governance, supply chain operation, procurement, human resources, finance and accounting), and are described by a set of properties. Among them, we include the unit of measurement, a textual description, its business object (e.g., Cost, Velocity), and the dimensions along which it can be computed (e.g., Time, Product, Organization) according to the multidimensional model.

For what concerns dimensions, they are arranged in a hierarchy of levels through a *partOf* relation, where each level represents a different way of grouping elements of the same dimension [26]. In this way, the “Month” level is partOf “Quarter”, while this is partOf “Semester”, that in turn is partOf “Year”. Instances of levels are called members (e.g. “2013” is member of “Year”), and are related to other instances through the same *partOf* relation. Through this property, and its inverse *hasPart*, it is possible to implement the classical roll-up and drill-down operators for analysis of multidimensional cubes, that allow to navigate among levels to aggregate and disaggregate values.

A KPI can be either atomic or compound, i.e. it can be computed from other lower-level indicators. In this case, dependencies of compound KPIs on its operands are defined by means of algebraic expressions, that is a *Formula* capable to express the semantics of an indicator. A formula describes how a compound indicator is calculated. In KPIOnto each formula is characterized by an aggregation function, the specification of how the formula is presented, the semantics (i.e., the mathematical meaning) of the formula, and references to its components, which are in turn (formulas of) indicators. For instance, *Costs_DeliverPhase*, that measures all costs related to delivery phase, is compound and has the following formula *Costs_Ship + Costs_Pack + Costs_OrPre + Costs_Deliv*, while *Costs_Ship* is atomic and has no associated formula. In Figure 1 an excerpt of KPIOnto is shown, in which the indicator “Costs_DeliverPhase” is described together with a portion of the Indicator taxonomy. Relations between the indicator and its dimensions are shown, including a fragment of the Dimension hierarchy. As discussed in the experimental Section, the relations among indicators that are encoded in formulas can be represented as a *formula graph*, where each node is an indicator, linked to the components of its formula through edges.

While the descriptive information of the ontology are represented in OWL2⁴, KPI formulas are directly represented as algebraic expressions and stored in a repository of formulas that relies on the W3C Recommendation MathML⁵ and the emerging standard OpenMath⁶. They are two XML-based standards widely used for rendering equations in web browsers. They provide a language to represent mathematical formulas capable, respectively, to describe the content/presentation of a generic formula and the semantics of mathematical objects. Hence, through such languages, each formula is fully represented together with its dependencies.

⁴ <http://www.w3.org/TR/owl2-overview/>

⁵ <http://www.w3.org/Math/>

⁶ <http://www.openmath.org/>

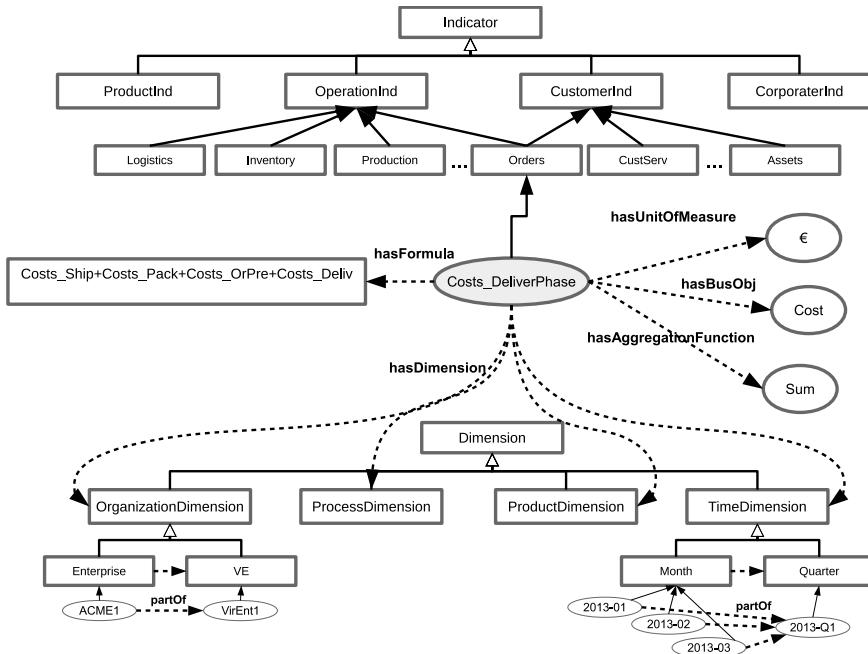


Fig. 1. Excerpt of KPIOnto

The knowledge available in KPIOnto is exploited to support advanced functionalities of the platform, like those described in next Section. In order to work with different typologies of languages and perform reasoning, we refer to Logic Programming as a common logical layer, for its capability to manipulate both OWL2 axioms and mathematical equations. In particular, reasoning functionalities have been implemented in Prolog, while XSB⁷ was chosen as logic programming database system for its efficiency. Basic functionalities are mainly aimed to enable manipulation of mathematical formulas, and targeted to achieve the following reasoning tasks:

- formula rewriting, simplification and resolution of equations, adapted from PRESS (PRolog Equation Solving System) [27], a formalization of algebra in Prolog. Such functions are also used to infer, for a given indicator, all the possible rewritings of a formula;
- checking of dependencies among formulas;
- multidimensional query handler, described in Subsection 4.1.

3.2 Raw Data Handler

The service platform and Raw Data Handler (RDH) are aimed to address data gathering and storage, together with the management of data interoperability

⁷ <http://xsb.sourceforge.net/>

Table 1. Example of table for Costs_Ship in the Data Storage

VE	Enterprise	Year	Semester	Quarter	Month	Week	Cat	Product	Value
VirEnt1	ACME1	2013	2013-S1	2013-Q1	2013-01	NULL	Armchair	NULL	6.23
VirEnt1	ACME1	2013	2013-S1	2013-Q1	2013-02	NULL	Armchair	Mod_245	1.29
VirEnt1	ACME2	2013	2013-S1	2013-Q1	NULL	NULL	Armchair	Mod_245	2.50
VirEnt1	ACME2	2013	2013-S1	2013-Q2	NULL	NULL	Armchair	NULL	13.22
VirEnt1	ACME2	2013	2013-S1	2013-Q2	NULL	NULL	Table	NULL	14.97

among the heterogeneous enterprises cooperating in a Virtual Enterprise. Data gathering and its semantic leverage are achieved through ETL modules, that extract all the relevant information from distributed data sources, and transform data items by providing the needed semantic enrichment. This last is performed by domain experts that annotate the schema of each source with respect to ontological concepts. It has to be noted that, although the mapping is done in a manual way, it is required only in the setup phase of the VE. Moreover, semantic-based tools are available in the ETL module to guide the user in the definition of mappings. Finally, the platform manages data loading into a Data Storage (DS), thus building a materialized centralized database that is updated whenever new data are made available by enterprises. In this work we refer to MySQL as database management system.

Each indicator described in KPIOnto is represented in the DS by a dedicated table, whose schema is defined by the indicator value, the specification of the enterprise and the VE, together with the related dimensions. It has to be noted that, in general, each enterprise provides data about only a subset of all possible indicators, hence resulting in a sparse DS. For what concerns dimensions, the schema contains a column for each level of every compatible dimension. For instance, a fragment of the schema for Costs_Ship is shown in Table 1, in which some example records for two different fictitious enterprises (ACME1, ACME2) are also represented. Each record shows a KPI value (in thousands of euros) that was measured at a different time and for a different product, while NULL values are reported whenever the KPI value is not specified for a certain level because it was pre-aggregated at a higher level (e.g. in the first, Week is NULL since the value refers to overall shipping costs measured in the whole 2013-01 and for the whole category Armchair, while the second refers to a specific product).

It has to be noted that several conflicts may occur while mapping enterprise data to KPI described in KPIOnto. In fact, besides usual conflicts about names or dimensions (e.g. homonymous indicators, or the same indicator measured along different dimensions), also conflicts about measures can occur (e.g., the same indicator calculated through different formulas). For lack of space we do not delve into further details on strategies to reconcile such situations, and refer the interested reader to [28], and to [24] for more general details about RDH.

4 KPI Explorer

In order to access KPI data, a uniform view over them is needed, independently of the original data sources. The KPI Explorer is a tool aimed to allow users to

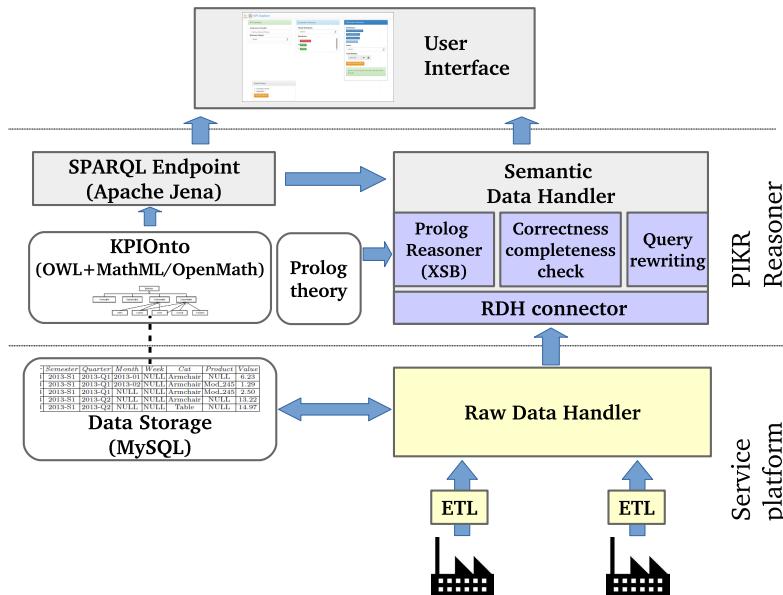


Fig. 2. Architecture of KPI Explorer and back-end services

visually compose a request expressed in terms of the common reference vocabulary for the platform, i.e. the KPIOnto, relying on back-end services that rewrite and resolve such requests as queries over the RDH.

Following the underlying model, a multidimensional query (MDQ) is posed by specifying the following information:

- the name of the requested indicator (e.g., Costs_DeliverPhase),
- for each compatible dimension, the specific level on which to aggregate (e.g., Cat for ProductDimension and Month for TimeDimension),
- an optional set of members on which to filter (e.g., {Armchair} for Cat and {2013-01,...,2013-12} for Month),
- the name of the requested VE and the names of the requested enterprises (e.g., ACME1 and ACME2),
- an optional flag specifying whether to obtain results for each enterprise separately or to aggregate them.

The output of a MDQ is rendered as a table containing all the results that are available in (or that can be dynamically computed from) the Data Storage.

From an architectural point of view, as shown in Figure 2, the KPI Explorer relies on a graphical user interface and on the following components:

- a) KPIOnto endpoint, a service relying on Apache Jena⁸ to query the ontology through the W3C Recommendation SPARQL language, in order to load from the ontology all the information to fill the fields in the GUI;

⁸ <http://jena.apache.org>

- b) Semantic Data Handler, the main back-end functionality that manages the execution of a multidimensional query on the Data Storage.

At first, (a) all the needed information about indicators and dimensions are retrieved from the ontology, in order to setup the user interface. After the user has composed a query, this is sent to the Semantic Data Handler (b), that manages its rewriting in SQL and execution over the Data Storage. Once obtained the results, the user can refine them by performing drill-down of either a level or the indicator. While the former is a typical operator in data warehouses, the latter enables a novel mode of exploring KPI data, allowing to re-execute the same query over the dependencies of the original requested KPI, as shown in the example in Subsection 4.2.

The following Subsections discuss in more details the Semantic Data Handler service and the GUI, and provide some examples.

4.1 Semantic Data Handler

The Semantic Data Handler (SDH) is a back-end functionality deployed as a web service within the PIKR, that handles multidimensional queries and embeds reasoning techniques for advanced search of KPI data. Starting from a MDQ expressed with the KPIOnto terminology, the SDH proceeds with its interpretation and checks its correctness, both syntactically (by verifying whether it is well-structured and valid with respect to an XML Schema) and semantically (e.g., by verifying if all terms are actually valid ontological concepts, or if every specified member belongs to the correct level).

Through a query rewriting mechanism, the SDH produces a new query compliant with the specific database management system⁹, that is then passed to the RDH for retrieving actual data.

However, in the context of the BIVEE project we do not assume the Data Storage to be complete. In fact, data are usually sparse because each enterprise provides only a subset of all indicators, and for each indicator only few values for every possible combination of members. Furthermore, there is no a priori agreement on the aggregation level for every dimension: e.g., one enterprise can provide data pre-aggregated on a monthly basis, while another on a quarterly basis, preventing the comparison of their performances.

To solve this kind of heterogeneity, an advanced feature of the SDH, namely *completeness check*, is devised to support the dynamic computation of those values that are not materialized, by exploiting the semantic description of the dimensions and the KPIs provided by KPIOnto. Such a functionality, for each result item required by the user (i.e. for every combination of members in the request) that is not available in the Data Storage, recursively applies a set of rewriting rules based on Logic Programming. To this purpose, the content of KPIOnto (indicator/dimension details and formulas) is translated as Prolog facts. Such rules allow to automatically (1) calculate an indicator from its formula (if the

⁹ Currently the RDH relies on SQL and MySQL, but the approach is valid for any alternative solution, like MDX queries on OLAP systems.

formula is not defined, the Prolog reasoner tries to infer it by manipulating and reverting already existing formulas), or (2) aggregate the value for a member of a level from its lower-level members.

To give an example of the two rules, let us consider the following MDQ:

$\langle \text{Costs_DeliverPhase}, \{\text{Cat}, \text{Quarter}\}, \{\{\text{Armchair}\}, \{2013\text{-}Q1\}\}, \{\text{ACME}_x, \text{ACME}_y\} \rangle$.

If for ACME_y there is no value for the Cat Armchair and for the Quarter 2013-Q1, the system tries to apply the rewriting rules. Through rule (1) by exploiting the formula $\text{Costs_DeliverPhase} = \text{Costs_Ship} + \text{Costs_Pack} + \text{Costs_OrPre} + \text{Costs_Deliv}$, a new query is automatically generated for each dependency of the KPI. In this case the new queries are:

$\langle \text{Costs_Ship}, \{\text{Cat}, \text{Quarter}\}, \{\{\text{Armchair}\}, \{2013\text{-}Q1\}\}, \{\text{ACME}_y\} \rangle$
 $\langle \text{Costs_Pack}, \{\text{Cat}, \text{Quarter}\}, \{\{\text{Armchair}\}, \{2013\text{-}Q1\}\}, \{\text{ACME}_y\} \rangle$
 $\langle \text{Costs_OrPre}, \{\text{Cat}, \text{Quarter}\}, \{\{\text{Armchair}\}, \{2013\text{-}Q1\}\}, \{\text{ACME}_y\} \rangle$
 $\langle \text{Costs_Deliv}, \{\text{Cat}, \text{Quarter}\}, \{\{\text{Armchair}\}, \{2013\text{-}Q1\}\}, \{\text{ACME}_y\} \rangle$

If such queries are satisfiable, then the four values are retrieved and aggregated by using the formula. The result is the value for $\text{Costs_DeliverPhase}$ requested by the original query.

An alternative solution, according to rule (2), is to drill-down member 2013-Q1 or member Armchair to their lower members¹⁰. This implies, for every member, to create a new query and (if and once solved) to compute the required value from those available by means of the aggregation function of the indicator (if any). In the example, if the system drills-down 2013-Q1 to its components 2013-01, 2013-02 and 2013-03, and if the aggregator for the KPI is “Sum”, then it is possible to calculate the value as a summation of the results of the following queries:

$\langle \text{Costs_DeliverPhase}, \{\text{Cat}, \text{Month}\}, \{\{\text{Armchair}\}, \{2013\text{-}01\}\}, \{\text{ACME}_y\} \rangle$
 $\langle \text{Costs_DeliverPhase}, \{\text{Cat}, \text{Month}\}, \{\{\text{Armchair}\}, \{2013\text{-}02\}\}, \{\text{ACME}_y\} \rangle$
 $\langle \text{Costs_DeliverPhase}, \{\text{Cat}, \text{Month}\}, \{\{\text{Armchair}\}, \{2013\text{-}03\}\}, \{\text{ACME}_y\} \rangle$

The result set obtained with the completeness check is always a superset of what can be generated without. However, due to its computational complexity, it is noteworthy that the running time of the completeness check procedure is related to the size of KPIOnto and the Data Storage. We refer the interested reader to [29] for a detailed discussion on the indicator drill-down rule.

4.2 User Interface

The front-end of the KPI Explorer enables users to graphically compose a MDQ with the aim to analyse KPIs for one or more enterprises. To present the main

¹⁰ Relations among members that are used in this case are the partOf introduced in Section 3.1.

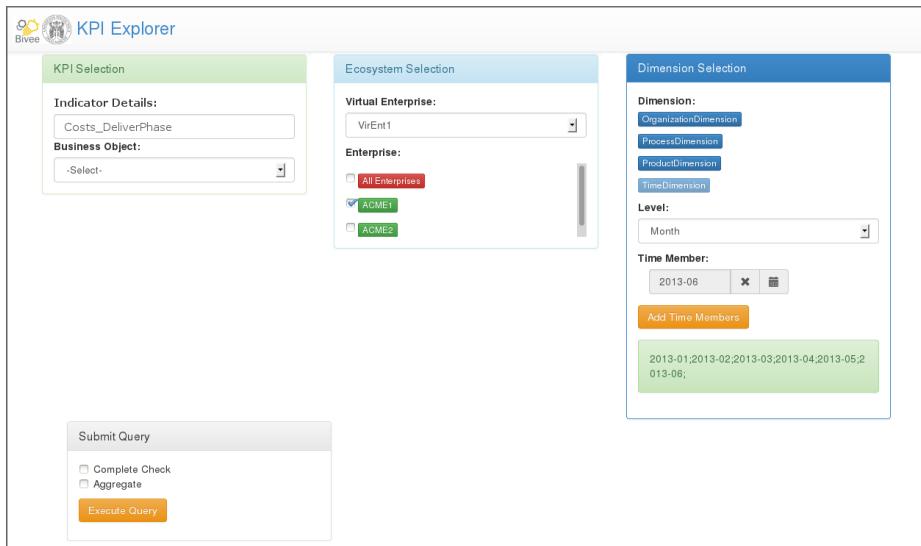


Fig. 3. KPI Explorer: composition of a query

Cat	Month	Enterprise	Costs_DeliverPhase
Armchair	2013-01	ACME1	15.7
Armchair	2013-02	ACME1	15.1
Armchair	2013-03	ACME1	15.3
Armchair	2013-04	ACME1	14.8
Armchair	2013-05	ACME1	13.9
Armchair	2013-06	ACME1	14.3

Fig. 4. KPI Explorer: result table

features, let us consider the following case study, in which a user wants to analyse the *Costs.DeliverPhase* for enterprise ACME1 belonging to the Virtual Enterprise *VirEnt1*.

As she specifies the required indicator within the search-box (see Figure 3), the rest of the form is dynamically loaded, including the list of compatible dimensions. In the example, the user decides to analyse the KPI with respect to *ProductDimension* and *TimeDimension*. For what concerns the former, she selects *Cat* level and specifies *Armchair* as member on which to filter the results. As for the latter, she specifies the first six months in 2013. Unselected dimensions are not part of the MDQ. After the execution of the query by the SDH, results are represented in a table, as shown in Figure 4.

The user notices a drop in the costs during spring 2013. In order to understand the reason behind this variability, she can refine the analysis by performing a drill-down of either a level or the KPI, by clicking on the “plus” symbol next to each label. As for the former (e.g., Month), the GUI retrieves from KPIOnto the lower level(s) (Week in this case), and performs a query at that

Cat	Month	Enterprise	Costs_DeliverPhase				
			Costs_Ship	Costs_Pack	Costs_OrPre	Costs_Deliv	
Armchair	2013-01	ACME1	15.7	6.23	2.15	3.1	4.22
Armchair	2013-02	ACME1	15.1	6.1		3.24	3.73
Armchair	2013-03	ACME1	15.3	6.03	2.23	3.09	3.95
Armchair	2013-04	ACME1	14.8				
Armchair	2013-05	ACME1	13.9	5.73	2.17	3.37	2.63
Armchair	2013-06	ACME1	14.3	5.5	2.19	3.13	

Fig. 5. KPI Explorer: drill-down of Costs_DeliverPhase

level. As regards the latter, the GUI calls the KPIOnto endpoint to retrieve the formula for *Costs_DeliverPhase*; then, for each of its components, a new query is created and sent to the SDH. Results about this last case, as shown in Figure 5, contribute to improve the user awareness about which specific sub-indicator mostly affected the final cost. To further deepen the analysis, the user performs a new drill-down on the sub-indicator *Costs_Pack*, as shown in Figure 6(a). Empty cells in the results stand for unavailable data, that have not been provided by the enterprise. For such a reason, the user executes again the last query enabling the “completeness check” option. As shown in Figure 6(b), for some of the previously empty cells now a value is shown (red boxes), inferred by using the rewriting rule (1) introduced in Section 4.1. In detail, to retrieve the value for *Materials_Pack* referring to 2013-01, the reasoner exploits the formula $Costs_Pack = Materials_Pack + Others_Pack + LabourCosts_Pack$ and solves it for *Materials_Pack*. To make an other example, the value for *Others_Pack* referring to 2013-05 is obtained as follows: at first the equation *Costs_DeliverPhase* was solved for *Costs_Pack*; then, the reasoner exploits the formula for *Costs_Pack* shown in the previous example.

As a second example, the user wants to compare performances about the same KPI for ACME1 and ACME2 for the first two quarters of 2013 and for Armchair category. However, the two enterprises provided different data with different granularity. In fact, ACME1 provided measures of both *Costs_DeliverPhase* and all its components *Costs_Ship*, *Costs_Pack*, *Costs_OrPre* and *Costs_Deliv*. On the other hand, ACME2 provided values for all of them, except for the *Costs_DeliverPhase*. Moreover, ACME1 measures values on a monthly basis, while ACME2 on a quarterly basis. Since no direct comparison can be made due to such heterogeneities, the user enables the “completeness check”. In such a way, the SDH exploits the Prolog-based rewriting rules discussed above. In more details, rule (1) is applied to calculate *Costs_DeliverPhase* for ACME2 starting from its dependencies, while rule (2) is exploited to aggregate months in quarters for ACME1. As a result, the user obtains the required KPI at the given levels, as shown in Figure 7.

Cat	Month	Enterprise	Costs_DeliverPhase			Materials_Pack	Others_Pack	LabourCosts_Pack	Costs_OriPre	Costs_Deliv
			Costs_Ship	Costs_Pack						
					Materials_Pack					
Armchair	2013-01	ACME1	15.7	6.23	2.15		0.1	1.31	3.1	4.22
Armchair	2013-02	ACME1	15.1	6.1		0.69		1.23	3.24	3.73
Armchair	2013-03	ACME1	15.3	6.03	2.23	0.62	0.63	0.98	3.09	3.95
Armchair	2013-04	ACME1	14.8			0.68	0.33	1.14		
Armchair	2013-05	ACME1	13.9	5.73	2.17	0.71		1.2	3.37	2.63
Armchair	2013-06	ACME1	14.3	5.5	2.19	0.72	0.4	1.07	3.13	

Cat	Month	Enterprise	Costs_DeliverPhase			Materials_Pack	Others_Pack	LabourCosts_Pack	Costs_OriPre	Costs_Deliv
			Costs_Ship	Costs_Pack						
					Materials_Pack					
Armchair	2013-01	ACME1	15.7	6.23	2.15	0.74	0.1	1.31	3.1	4.22
Armchair	2013-02	ACME1	15.1	6.1		0.69	0.11	1.23	3.24	3.73
Armchair	2013-03	ACME1	15.3	6.03	2.23	0.62	0.63	0.98	3.09	3.95
Armchair	2013-04	ACME1	14.8			0.68	0.33	1.14		
Armchair	2013-05	ACME1	13.9	5.73	2.17	0.71	0.26	1.2	3.37	2.63
Armchair	2013-06	ACME1	14.3	5.5	2.19	0.72	0.4	1.07	3.13	

Fig. 6. KPI Explorer: drill-down of Costs_Pack (a) without and (b) with completeness check

Submit Query
 Complete Check
 Aggregate

Execute Query
 Show Query

Output

Cat	Quarter	Enterprise	Costs_DeliverPhase
Armchair	2013-Q1	ACME1	46.1
Armchair	2013-Q2	ACME1	43
Armchair	2013-Q1	ACME2	35.27
Armchair	2013-Q2	ACME2	34.8

Fig. 7. KPI Explorer: results for ACME1 and ACME2 with completeness check

5 Evaluation

The goal of this Section is to provide an evaluation of costs and benefits of the proposed approach, respectively in terms of efficacy and response time.

As for the former, we evaluated how larger it is the number of indicators that can be dynamically computed through the reasoning mechanism, with respect to those directly available in the Data Storage. To evaluate how the result depends on the size of the ontology, the experimentation was performed on a set of synthetic ontologies with an increasing number of indicators, and then on the real-world ontology used in the BIVEE project. Synthetic ontologies were au-

tomatically generated by varying two parameters of the corresponding formula graph: the number of operands per indicator (op), and the maximum number of levels (lev). For sake of simplicity, all formulas for the indicators are defined as summation of other randomly chosen indicators. For instance, a 1-level ontology contains only one top indicator that is computed by summing op indicators; a 2-levels ontology is formed by one top level indicator, which is computed on op indicators in the middle layer that, in turn, are computed on op low-level indicators. The number of indicators in each ontology is given by $\sum_{i=0}^{lev} op^i = \frac{1 - op^{lev+1}}{1 - op}$, ensuring that each indicator has exactly op number of operands. Table 5 reports the list of generated ontologies with the number of indicators.

Table 2. Size of the generated ontologies: operands, levels and number of indicators

op	2	3	4	5	2	3	2	2
lev	2	2	2	2	3	3	4	5
#indicators	7	13	21	31	15	40	31	63

Let F_i be the set of all possible inferable formulas for the indicator i of the given ontology O ; the procedure followed in the experimentation is:

For $k = 1$ to 100

1. randomly define a subset S of indicators in O
2. for each indicator $i \in O$
check whether i is computable over S

An indicator i is *computable* over the set S if at least a formula $f \in F_i$ exists such that all operand of f are indicators in S , hence f can be computed by using only available indicators. Given that results depend on the number of indicators actually materialized, for each ontology the procedure is executed multiple times, namely 100: each time a randomly selected subset of indicators in the ontology is assumed to be actually available in the Data Storage.

In order to keep the complexity of the experimental procedure under control, for each indicator, the maximum number of formulas the reasoner inferred was set to 50. Figure 8 shows the rate of computable indicators (i.e. the ratio between the number of computable indicators and the number of indicators in the ontology) vs. the rate of available indicators in Data Storage (i.e. the ratio between the number of available indicators in the Data Storage and the number of indicators in the ontology). For each ontology we drew a different line in the figure. The bisector (dotted line) represents the baseline, where the number of computable indicators is equal to number of those available. In this case only indicators that are actually available in the Data Storage can be computed, i.e. points on the bisector represent the situation where no reasoning is performed. We'd like to note that all results are above the bisector, demonstrating the efficacy of the approach. When few indicators are available (i.e. the rate of available

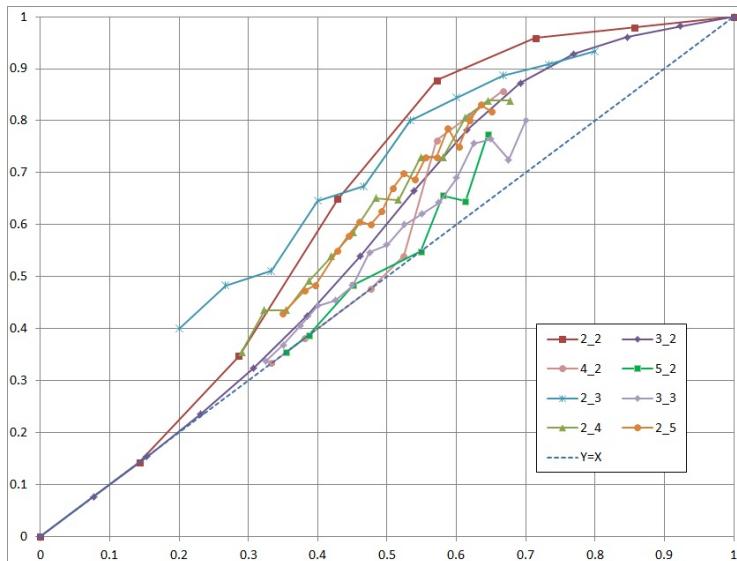


Fig. 8. Rate of computable indicators vs. rate of available indicators in Data Storage

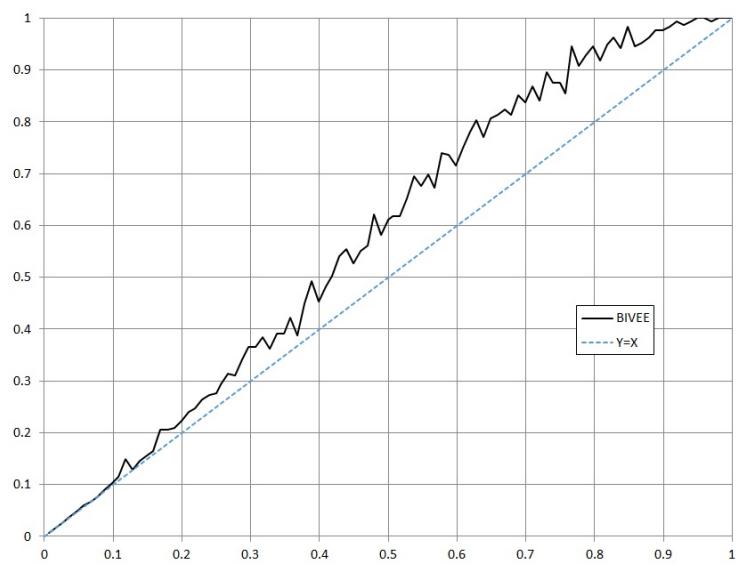


Fig. 9. Rate of computable indicators vs. rate of available indicators in Data Storage, for KPIOnto

indicators is less than 0.15), the approach does not outperform the baseline, hence the reasoning mechanism is ineffective. On the other side, with a rate of available indicators higher than 0.65, we are able to compute more than the 80% of indicators in the ontology. The same remarks can be made also for results obtained over KPIOnto, the real-world ontology used in the BIVEE project, as shown in Figure 9. This ontology takes into account 356 KPIs and its formula graph is formed by a set of disconnected graphs, with different levels (from 1 to 5) and operands (from 2 to 4). On average, the graph has 3.14 levels and 2.67 operands per indicator.

Table 3 shows the average execution time of the reasoning mechanism for inferring all formulas for an indicator. Experiments were carried on an Intel i7 CPU 2.20GHz with 8GB memory, running Windows 7 64bit. For some of the ontologies (e.g., lev=3 and op=4), it was not feasible to conduct the tests. We'd like to note that the execution time does not increase with the number of indicators, but it depends on the topology of the ontology, i.e. the way in which the indicators are connected to each other. On average, the inference of all formulas required less than 1.3s. As regards KPIOnto, the execution time is on average around 6s. However, it is noteworthy that real scenarios do not usually involve extracting the complete set of formulas for an indicator, reducing in this way the overall execution time.

Table 3. Average execution times for inferring all formulas for an indicator (ms)

Op	Lev			
	2	3	4	5
2	69.43	211.69	651.87	2083.38
3	528.93	1699.63	3792.43	n/a
4	1294.05	n/a	n/a	n/a
<i>Average</i>		1291.43		
<i>KPIOnto</i>		5968.14		

6 Conclusion

In this paper, we presented a tool for exploration of data related to Key Performance Indicators, provided by multiple enterprises collaborating in a shared innovation project, in the context of a Virtual Enterprise. The tool relies on the architecture of the BIVEE platform, and in particular on a semantic middleware that includes both a domain ontology and a repository where heterogeneous data are annotated and stored. Back-end functionalities of the tool ensure the rewriting of the user query, expressed in ontological terms, and its execution on the Data Storage. Advanced functions enable automatic, dynamic and transparent aggregation of KPIs at any level, and computation of KPI values through their mathematical formulas, by exploiting information in the ontology. As for the front-end, such an approach allows for a novel mode in data exploration, since

users can expand a KPI into its more basic components, thus enabling the possibility to deepen the analysis by browsing the KPI hierarchy, in an effective and efficient way, as shown by the encouraging outcome of the experimentation. Future work will focus on providing more detailed and formal descriptions of the underlying reasoning functionalities, and on performing more comprehensive tests to evaluate the applicability of the proposed approach in real-life scenarios.

Acknowledgments. This work has been partly funded by the European Commission through the FoF-ICT Project BIVEE (No. 285746). The authors wish to thank the Commission for its support and all the project partners for their contribution in the development of various ideas and concepts presented in this paper.

References

- Chesbrough, H.: Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business Press, Boston (2003)
- Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., Turricchia, E.: Olap query reformulation in peer-to-peer data warehousing. *Information Systems* 37, 393–411 (2012)
- Torlone, R.: Two approaches to the integration of heterogeneous data warehouses. *Distrib. Parallel Databases* 23, 69–97 (2008)
- Priebe, T., Pernul, G.: Ontology-Based Integration of OLAP and Information Retrieval. In: Proc. of DEXA Workshops, pp. 610–614 (2003)
- del Río-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortés, A.: On the definition and design-time analysis of process performance indicators. *Information Systems* 38, 470–490 (2013)
- Lee, C., Chen, C.J., Lu, H.: An aspect of query optimization in multidatabase systems. *SIGMOD Rec.* 24, 28–33 (1995)
- Lenzerini, M.: Data integration: A theoretical perspective. In: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2002, pp. 233–246. ACM, New York (2002)
- Cruz, I.F., Xiao, H.: The role of ontologies in data integration. *Journal of Engineering Intelligent Systems* 13, 245–252 (2005)
- Lakshmanan, L.V.S., Pei, J., Zhao, Y.: Efficacious data cube exploration by semantic summarization and compression. In: VLDB, pp. 1125–1128 (2003)
- Neumayr, B., Anderlik, S., Schrefl, M.: Towards Ontology-based OLAP: Datalog-based Reasoning over Multidimensional Ontologies. In: Proc. of the Fifteenth International Workshop on Data Warehousing and OLAP, pp. 41–48 (2012)
- Niemi, T., Toivonen, S., Niinimäki, M., Nummenmaa, J.: Ontologies with semantic web/grid in data integration for olap. *Int. J. Sem. Web Inf. Syst.* 3, 25–49 (2007)
- Huang, S.M., Chou, T.H., Seng, J.L.: Data warehouse enhancement: A semantic cube model approach. *Information Sciences* 177, 2238–2254 (2007)
- Xie, G., Yang, Y., Liu, S., Qiu, Z., Pan, Y., Zhou, X.: EIAW: Towards a Business-Friendly Data Warehouse Using Semantic Web Technologies. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 857–870. Springer, Heidelberg (2007)

14. Pedrinaci, C., Domingue, J.: Ontology-based metrics computation for business process analysis. In: Proc. of the 4th International Workshop on Semantic Business Process Management, pp. 43–50 (2009)
15. Kehlenbeck, M., Breitner, M.: Ontology-based exchange and immediate application of business calculation definitions for online analytical processing. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 298–311. Springer, Heidelberg (2009)
16. Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., Mylopoulos, J.: Strategic business modeling: representation and reasoning. Software & Systems Modeling (2012)
17. Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. Information Systems 35, 505–527 (2010)
18. del-Río-Ortega, A., Resinas, M., Ruiz-Cortés, A.: Defining process performance indicators: An ontological approach. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 555–572. Springer, Heidelberg (2010)
19. Diamantini, C., Potena, D.: Semantic enrichment of strategic datacubes. In: Proc. of the ACM 11th International Workshop on Data Warehousing and OLAP, DOLAP 2008, pp. 81–88 (2008)
20. Diamantini, C., Potena, D., Proietti, M., Smith, F., Storti, E., Taglino, F.: A semantic framework for knowledge management in virtual innovation factories. Int. J. Inf. Syst. Model. Des. 4, 70–92 (2013)
21. Roubioni, N.: Hypertext glossary of business cycle indicators,
<http://people.stern.nyu.edu/nroubini/bci/bci.html>
22. Crosta, F.: Indicatori di performance aziendali: come definire gli obiettivi e misurare i risultati (in italiano). Franco Angeli (2005)
23. Supply Chain Council: Supply chain operations reference model. Supply Chain Council (2008)
24. Isaja, M., Diamantini, C., Potena, D., Storti, E., Taglino, F., Smith, F.: BIVEE integrated semantic meta-space: advanced interoperability services. Deliverable 5.3. Bivee European project (2013)
25. Diamantini, C., Potena, D., Storti, E.: A logic-based formalization of KPIs for virtual enterprises. In: Franch, X., Soffer, P. (eds.) CAiSE Workshops 2013. LNBP, vol. 148, pp. 274–285. Springer, Heidelberg (2013)
26. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd edn. John Wiley & Sons, Inc., New York (2002)
27. Sterling, L., Bundy, A., Byrd, L., O’Keefe, R., Silver, B.: Solving symbolic equations with press. J. Symb. Comput. 7, 71–84 (1989)
28. Diamantini, C., Potena, D., Storti, E.: Data mart reconciliation in virtual innovation factories. In: Iliadis, L., Papazoglou, M., Pohl, K. (eds.) CAiSE 2014 Workshops. LNBP, vol. 178, pp. 274–285. Springer, Heidelberg (2014)
29. Diamantini, C., Potena, D., Storti, E.: Extending drill-down through semantic reasoning on indicator formulas. In: Bellatreche, L., Mohania, M. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 57–68. Springer, Heidelberg (2014)

Arabic-English Domain Terminology Extraction from Aligned Corpora

Wiem Lahbib¹, Ibrahim Bounhas^{1,2}, and Bilel Elayeb^{3,4}

¹LISI Laboratory of computer science for industrial systems, Carthage University, Tunisia
wiemlahbib88@hotmail.fr

²Higher Institute of Documentation (ISD), Manouba University, 2010 Tunisia
Bounhas.Ibrahim@gmail.com

³RIADI Laboratory,The National School of Computer Science (ENSI),Manouba University
2010 Tunisia

⁴Emirates College of Technology, P.O. Box: 41009. Abu Dhabi, United Arab Emirates
Bilel.Elayeb@riadi.rnu.tn

Abstract. The rapid growth of information sources has produced a large amount of electronically stored documents evolving every day. The development of Information Retrieval Systems (IRS) is a response to this growth, which aims to help the user identify relevant information. Recent IRS proposes to guide the user through providing domain knowledge in the form of controlled vocabularies or terminologies and thus, domain ontologies. In this context, it is necessary to develop multilingual termino-ontological resources. This paper proposes a new approach for bilingual domain terminology extraction in Arabic and English languages as a first step in the bilingual domain ontology building, to be exploited in terminological search. The approach uses arabic vocalized texts to reduce ambiguities and the alignment process to extract the english translations. To the best of our knowledge, the process implemented in our approach (morphological analysis, arabic terminology extraction, alignment and extraction of english translations) is the first work in the field of arabic-english bilingual domain terminology extraction. The results of experiments are encouraging showing rates of relevant term extraction from multiple domains and their translations exceeding 89%.

Keywords: Bilingual terminology, bilingual ontology, alignment process, parallel corpora, terminological research.

1 Introduction

Bilingual terminology extraction is an important task many applications, such as bilingual ontology building and machine translation. Nevertheless, related work in the field of Arabic domain terminology extraction is limited compared to other fields, such as arabic monolingual terminology extraction [1], arabic ontology construction [2], Machine Translation [3], comparable articles identification [4], and parallel sentences extraction [5]. This discrepancy is explained by the lack of parallel corpora of high quality and/or the complexity and scarcity around the alignment task of arabic

corpora. In fact, this task is difficult due to the specific characteristics of Arabic as a derivational and inflectional language. Indeed, it is a rich language and nuances in words are numerous, which makes hard to develop efficient tools which deal the ambiguity and the richness of Arabic texts in IR and terminology extraction.

Our project aims to develop a multilingual semantic portal, providing opportunities for non-native Arabic speakers in searching in arabic collections, using arabic or english queries. The ability to go beyond the language of the collection, allows the user to move in all possible directions to explore the multilingual collection and not be limited to documents expressed in just one language. This allows limiting the silence and increasing the recall. One of the searching modes provided to guide users in their activities is based actually on bilingual terminologies.

In addition, extracting specific bilingual terminologies is very important and crucial for what concerns the building of bilingual ontology.

This paper presents a hybrid approach that extracts a domain terminology from Arabic-English (AR-EN) corpora represents the first step of domain ontology building. This approach is mainly based on the alignment process and represents an attempt to improve the extraction of terminologies with the statistical analysis and improving outputs provided by the alignment tool. The input carry text fragments (AR-EN) analyzed using morphological disambiguation tools. The result of the alignment will be used to extract couples of terms from Arabic and English that will be exploited in the building of bilingual domain ontology.

This document is organized as follows. The second section presents a literature review. The proposed approach is presented in section 3. In the fourth section, we present the test collection and its main characteristics. The fifth section details and interprets our experimental results. Finally, we criticize our work and propose some directions for future research in the conclusion.

2 Related Works

This section provides an overview of the main existing approaches of bilingual terminology extraction. The objective is to study how couples of relevant terms can be extracted from a representative collection of documents. These approaches can be organized according to whether they are applied to a specific domain or not.

2.1 A Specific Domain-Based Approaches

These approaches use a domain-specific corpus for extracting terminology. The proposed approach of [6] aims to extract candidate terms by applying linguistic models specific for English and Spanish languages. The alignment process is applied at the sentence level. This approach has been applied to the field of medicine, leading to an accuracy of translation equal to 80%. However, the sentence alignment was completely manual which influenced the results. In the same orientation Bouamor and al. [7] have presented an approach of bilingual (French-English) lexicons extraction from the financial and medical fields. They tried to improve the standard approach (construction,

transfer and comparison of context vectors) by performing a lexical disambiguation of context vectors in order to solve the problem of polysemous words revealed from the used bilingual dictionary. To do this, they used the WordNet thesaurus allowing the derivation of the semantic similarity between the elements of each vector. Although this approach has completed an accuracy of 67.1% in the field of breast cancer with +3.4% as gain, it obviously depends on the level of the specificity of the used corpus. In a second experiment, Bouamor and al. [8] have used Wikipedia to create a dictionary of bilingual translation. The authors have begun their approach by deriving a list of similar Wikipedia concepts for each given word to be translated and its context vector in the source language, using the ESA (Explicit Semantic Analysis) inverted index [9] using TF-IDF. Then they have used a translation graph to locate the corresponding concepts in the target language. The graph was extracted from translation links of Wikipedia articles. Once the translations are found, the bilingual dictionary is built. This approach was experimented on different areas: corporate finance, breast cancer, wind energy and mobile technology; and against two pairs of languages: French-English and Romanian-English. They have reached an F-Measure of 0.81 on the top 20 words in the breast cancer field. For what concern the dictionaries, the resulting translation dictionaries contain 193,543 entries for French-English and 136,681 entries for Romanian-English. However, the created dictionaries are not complete since Wikipedia doesn't cover all domain relevant words of one language. Besides lexical disambiguation, Hazem and Morin [10] have combined the two concepts of contextual representation - graphical representation and representation of syntactic dependencies - to identify translations using the French-English dictionary ELRA-M0033. Indeed, it consists on merging the two lists of scores resulting then from each representations of each word to be translated through an arithmetic combination of scores. Two application areas namely - breast cancer and renewable energy - have been exploited. The approach of [11] has targeted the field of physics to extract bilingual terminology which is based on graph-based label propagation. Indeed, the principle of their technique is to transfer labels from labeled data to unlabeled data and so authors have captured co-occurrence relations by representing each word as a vertex in a graph, in addition to the exploitation of similarity graphs in the propagation process. Experiments were performed using English and Japanese patent documents. However, this approach has neglected word sense disambiguation problem.

Weller and al.'s approach [12] was experimented against English and German languages on the field of wind energy. After conducting a full linguistic analysis followed by the extraction of candidate terms based on linguistic patterns, the alignment of terms was performed using a bilingual dictionary. The latter is enriched by computing the similarities between words in both languages and inferring new translation patterns. The authors reported a precision of translation between 50% and 78%. The major concern with this approach is that it requires a bilingual dictionary, which is not available for all pairs of languages.

2.2 A General Domain-Based Approaches

There are several approaches for extracting bilingual terminologies that are not based on a specific area. Simoes and Almeida [13] were interested in the English and Portuguese

languages. They described translation patterns generated from a matrix of probabilistic translation extracted from NA Tools dictionary¹. The phenomenon of variations in translation has been taken into account when calculating these probabilities by adding morphological restrictions. They used a spell checker for English called jSpell allowing writing some additional inference rules, in order to improve the analysis of the Portuguese language. They used EuroParl² corpus to extract simple and compound words and they reached an accuracy rate of 96%. Lu and Tsou [14] have attempted to solve the problem of alignment by combining between Moses and GIZA++ toolkit [15] during the treatment of both Chinese and English languages. In fact, after completing the sentences alignment, they calculated the probability of translation and chose the most probable translation for each term in the source language. This approach has a high complexity caused by an alignment in two steps by two tools. Also, a large number of filters may introduce a risk of data loss. The approach of [16] introduced the concept of local frequency (in the sub-text that contain the term) and global frequency (in the entire corpus) in order to extract the right translations. In fact, after applying grammar rules during the first step, the frequency (local/global) should be greater than a predefined threshold. The results have given a precision of 77% using Dutch-English corpus with 25,000 Dutch words. However, Eijik's method has neglected the morphological analysis.

The volume of researches bearing on Arabic bilingual terminologies extracting is limited due to the complexity and high ambiguity that accompanies the alignment process. Sellami and al. [17] have proposed to extract a bilingual French-Arabic terminology and another French-Yoruba terminology. They applied their approach on Wikipedia and exploited cross-language links that connect two items whose titles are the translations of each other. Then, they proceeded to the alignment step using the Giza++ tool. In the assessment step, they opted for an expert-based validation and reported a precision between 0.46 and 0.74. However, one major weakness of this approach; it is limited to the extraction of simple terms from the titles of articles. The content of these articles is not analyzed, thus reducing the recall.

2.3 Existing Works Summary

The above detailed state-of-the-art allows us to note the following facts:

- First, existing works on extracting domain bilingual terminologies including Arabic are rare and limited in scope. To the best of our knowledge, the approach of [17] has been the only work in this field. But, it has not implemented sophisticated methods for linguistic analysis and statistical content. It also neglects the semantic distribution of terms.
- Arabic Wikipedia is not vocalized, thus resulting in many types of morphological ambiguity. Furthermore, it is not segmented into domains. To improve the relevance, we have to compare the distribution of terms in different fields. Besides, the complexity of the alignment process requires the introduction of manual intervention

¹ <http://linguateca.di.uminho.pt/natools/>

² <http://www.isi.edu/koehn/europarl/>

which is time consuming and makes development difficult to achieve fully automatic approaches.

- Finally, we need high quality Arabic corpus segmented on different areas. This need becomes critical for certain language pairs for which we lack structured corpus or exploitable bilingual dictionaries.

Based on this discussion, we propose a hybrid approach combining linguistic and statistical analysis, but it differs from existing ones. Whereas they are limited to direct outputs provided by the alignment tool, we believe that improving results is essential to strengthen the final result. In order to reduce the morphological ambiguities and improve the alignment process, we use a vocalized corpus. This corpus is already segmented into areas to help studying the relative distribution of terms. In fact, our main goal in this paper is to study the possibility of using these techniques on AR-EN corpus, to capture the characteristics and constraints of Arabic language in the process. Our study focuses on the hadith corpus characterized by a number of specificities challenging terminology extraction systems.

3 The Proposed Approach

We present in this paper an approach that seeks to improve the extraction of bilingual terminology applied on Arabic and English corpora by combining statistical calculus and linguistic knowledge, thus taking advantage of the benefits and reducing the limits of these techniques. It is a corpus based approach, as we don't have specialized dictionaries, which can be directly used to improve alignment task. We focus on the classical Arabic, as many of his texts are vocalized.

The main objective of our approach is to extract AR-EN domain terminology. Consequently, we need to study how the alignment process and tools can be applied against the Arabic language, and what are the main difficulties since there is no such a work that have used such resources (language, technique and tools).

Figure 1 shows the overall architecture of our terminology extraction process. This process is completed in the following steps:

- Morphological analysis and disambiguation of each word of a given AR-EN corpus.
- Extraction of relevant Arabic terms of each area using basically statistical analysis.
- Translations are found based on a translation matrix generated from the alignment process, useful to extract bilingual domain terminology.

3.1 Morphological Analysis and Disambiguation

As we have mention above, the first step of our approach consists on morphological analysis and disambiguation of each word in the corpus. This step provides all possible solutions and values of the word's morphological features and its attributes such as Part-Of-Speech (POS), gender, number, etc. These attributes are exploited for the next steps especially extracting relevant terms.

In the following an example of an output of the morphological analysis and disambiguation:

الْخَمْرُ, NOUN, NCONJ, NPART, DET, NPROPN, NPERSN, NVOICE, NASPECT, NGENDER, S, NOM, NPREP, NMODE, NADJ, لِلْخَمْرِ
This example analyse the word “الْخَمْرُ” which means alcohol. There are 16 attributes including:

- Attribute 2 POS: NOUN (means the word **الْخَمْرُ** is a noun).
- Attribute 3 CONJUNCTION: NCONJ (means the word **الْخَمْرُ** isn't a conjunction).
- Attribute 5 DETERMINER: DET (means the word **الْخَمْرُ** is defined).
- Attribute 13 PREPOSITION: NPREP (means the word **الْخَمْرُ** isn't a preposition), etc.

The present work uses the probabilistic disambiguation tool of [18], which was developed in order to manipulate traditional Arabic texts. This tool allows to segment a sentence into words, to extract the lemma of each word and to remove inflectional elements (antefixes, prefixes, suffixes, post-fixed). Thereafter, it identifies the POS tagging. To treat the English side of the corpus, we integrated the Tree Tagger tool [19].

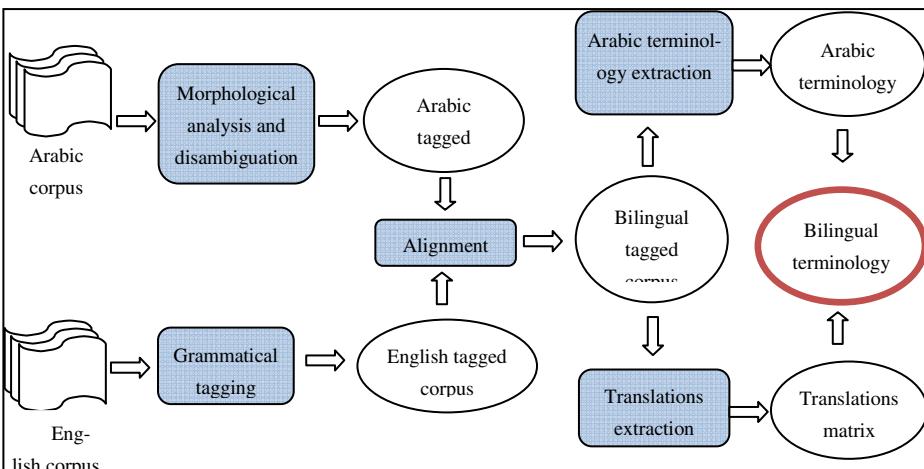


Fig. 1. The main steps of the proposed approach

3.2 Arabic Terminology Extraction

The second step is about extracting arabic domain terminology. Our terminology consist of single word nouns (POS=NOUN) assessed as relevant to the domain. To do this, we use the POS to filter other Part-Of-Speech on the one hand. On the other hand, to measure the relevance toward one domain, we use TF-IDF (Term Frequency-Inverse Document Frequency) [20]. The principle of this measure is to assess the importance of a word against different domains belonging to the corpus. This choice is mainly explained by two facts; first, it allows the comparison of words having too different frequencies unlike other measures such as the measure of Lafon [21] that needs having substantially equal compared parts. In addition, TF-IDF measure is very often used in Arabic corpora.

Based on the results, and after several changes and a set of threshold values, we chose 0.001 as the threshold that gives the best equilibration (noise/silence). Then, the words deemed relevant are those who have had a TF-IDF greater than this value. After sorting these words according to their TF-IDF scores in a descending order, we obtained the list of terms representing each area.

In this case, and if we consider the word **الحَمْرُ** manipulated in the previous phase, this word presents a NOUN, so it can be judged as domain relevant only if it comes with a $\text{TF-IDF} > 0.001$.

3.3 Alignment

The parallel alignment is a process of associating each sentence from a source language L1 with the sentence that represents the translation in the target language L2. In this context, the alignment process presents the main object of the third step of this work, but at the word level. To establish this process, some data preparation must be done first. To conduct our experiments, bi-texts (sentences) are generated by segmenting the set of documents representing a domain. The following sentence is considered as a bi-text:

“قالت عائشة لقد خرجنا مع رسول الله في بعض أسفاره”

(Aisha said we set out with Allah's Apostle on one of his journeys).

Then, we use the GIZA++ tool to align the AR-EN bi-texts at the word level. Thus, for each arabic word, the tool assigns a set of english words identified by their exact positions in the english corpus.

3.4 Translations Extraction

We reach, finally, the translations extraction phase. This step consists in extracting the most likely corresponding translation, for each arabic word. In this step, the input consists of the alignment file generated by GIZA++ on the one hand, and AR-EN corpus after being segmented and stemmed, in the other hand. Therefore, each lemma is assigned to different translations. Meaning that the word “**الحَمْرُ**” for example, can be translated through different translations that exist in the target language, such as alcoholic drink or alcohol or booze or wine, etc. In this situation, we must choose the most convenient translation as we explain in the next paragraph.

Because the translation of a word changes from one sentence to another, we decided to build a translation matrix to identify the correct translations. In fact, it is a matrix of N rows and M columns; where N is the number of different pairs of translations and M is equal to four (Arabic term, candidate translation, POS, pair's frequency). Our matrix consists of 5568 rows, meaning, 5568 candidate pairs of translations. This allowed the use of the co-occurrence of each pair. Therefore, translation consists of list of English words with their POS classes. The number of occurrences is the number of times in which the arabic word is aligned to the set of words that make up the english translation. Then, we select the most common translation with the highest frequency for each term.

4 Test Corpora

To apply our approach, a parallel corpus segmented into domains is needed. For that purpose, we used a corpus composed of Arabic stories called hadith. This corpus is freely available in many online encyclopedias³.

This choice is justified by many reasons. First, this corpus is translated into different languages such as English. Second, the hadith corpus is well structured, easily accessible and gives the opportunity to query by different facets (search by book, by domain, by narrator, etc). This structure will certainly help us extract bilingual domain terminology. Third, this corpus is a large encyclopedia, which covers an important period of the arabic history. Fourth, several versions of the corpus are vocalized, which reduces the ambiguity caused by the lack of diacritics in other corpora (such as Wikipedia). For all these reasons, it has been exploited in many application areas (e.g. semantic relation extraction [22], terminology extraction [1], reliability evaluation [23], ontology construction [2] and [24]).

We note that the specific characteristics of this corpus do not limit the ability of generalizing our approach using more conventional specialized texts. Finally, it is crucial to develop new technologies allowing the access to hadith literature which interests a large community. Thus, this corpus is suitable for our approach of extracting domain terminology.

Our experiments were performed in thirteen domains. This is where we must mention that hadith corpus is already structured on many domains which we have randomly chosen thirteen domains to conduct our experimentations. Table 1 presents some statistics about each domain. In the whole, the test corpus contains 23727 Arabic words and 45350 English words from 3187 pairs of sentences.

Table 1. Statistics about the test corpora

Domain	#Words	
	Arabic	English
The beginning of the revelation	بدء الوحي	1331 2548
Fasting	الصوم	3479 6910
Drinks	الأشربة	1642 3156
Faith	الإيمان	2026 3529
The minor ablution	الوضوء	3989 7450
The major ablution	الطهارة	1330 2481
The dry ablution	التيمم	1091 2071
Taraaweeh ⁴	التراويح	0691 1292
Prayer in seclusion	الإعكاف	0912 1704
Foods	الأطعمة	3599 6719
Al-Aqqa (Sacrifice of atonement)	القيقة	0209 0438
Sacrifices	الأضاحي	1111 2157
The sacrificed animals and hunting	ذبائح و الصيد	2317 4895
Total	23727	45350

³ See for example the IslamWeb encyclopedia (<http://library.islamweb.net/hadith/index.php>) which contains 1400 books.

⁴ A specific type of islamic collective prayer organized in Ramdhan

5 Experimentation and Interpretation

The current section presents and discusses the results obtained for Arabic terminology and translation extraction. Table 2 shows the total number of extracted terms for the selected domains. Besides, we opted for expert-based evaluation, thus computing the precision rates $P(\text{Term})$ which consist on calculating the ratio between the number of extracted terms judged as relevant and the number of total extracted terms. The same method is used to compute the accuracy of translation $P(\text{translation})$ which consist on calculating the ratio between the number of extracted pairs of translation judged as relevant and the number of total extracted translation couples. So, in order to assess the quality of our approach, we opted for an expert evaluation to calculate the accuracy rate by building a reference list for each domain comprising pairs of terms. The same method is used to calculate the accuracy of the translation. Unfortunately, it is difficult to compare our work against a baseline or existing works, due to differences between the corpora, the languages, the domains of application and/or the used resources. In addition, this comparison requires a standard test collection, which does not exist.

By analyzing table 2, it is clear that the number of terms does not depend on the corpus size. In fact, the biggest number of terms was obtained with “The beginning of the revelation” domain which is not the bigger one.

Table 2. Expert-based evaluation of terminology and translation extraction

Domain	#terms	P(term)	P(trans)
The beginning of the revelation	236	80%	90%
Fasting	150	82%	93%
Drinks	170	79%	90%
Faith	198	74%	92%
The minor ablution	140	84%	93%
The major ablution	140	86%	66%
The dry ablution	173	75%	92%
Taraaweeh	104	80%	90%
Prayer in Seclusion	100	80%	89%
Foods	086	85%	92%
Al-Aqqa	062	85%	90%
Sacrifices	135	72%	93%
The sacrificed animals and hunting	162	80%	90%
Average		80%	89%

By examining table 2, we notice that accuracy rates are close to 90%, however, multiples factors must be taken in consideration more deeply in the whole process of extraction to be improved and to be transitioned into practice. In fact, although these rates are encouraging, the results of Arabic terminology extraction are influenced by the semantic ambiguities. Indeed, some arabic terms in this corpus are closely related and some of them are polysemous. Besides, some domains are inter-dependent. For example, “fasting” (which requires stopping eating) is related to the domain of “foods”.

For translation extraction, we faced some problems related to the nature of the corpus, as some english expressions that are not exact translations of corresponding arabic ones. In other words, the human translators used the general meaning of the arabic hadiths to produce the english side of the corpus. This is critical for some classical terms, which have not direct translations in modern Arabic. Besides, GIZA++ is a probabilistic tool, generating some errors in the alignment process, which is not 100% reliable.

Figure 2 shows an example of an intermediate output generated after the alignment process. It shows that the translations frequencies are almost equivalent in most cases, making it difficult to choose the best translation.

Arabic word	Translation sequence	Frequency
خباء	tent/NN for/IN	2
خباء	tent/NN	2
خباء	notice/VVN the/DT tent/NNS	1
خباء	he/PP saw/NN	1
خباء	and/CC	1
سوط	a/DT whip/NN	1
سوط	my/PP\$ whip/NN	2
سوط	the/DT lash/VV	2
رمح	hand/NN him/PP his/PP\$ spear/NN	1
رمح	the/DT spear/NN	1
رمح	spear/NN	1
سعى	chase/VVD	1
سعى	I/PP run/VVD	1
سعى	chase/VVD it/PP	1

Fig. 2. Examples of translations before stop-word removal

To solve this problem, removing stop-words in the list of translations is required. Table 3 provides a description of English stop-words accompanied by some examples. Then, the number of occurrences of each translation is recalculated. This produces a new set of translation vectors, as shown in Figure 3.

Table 3. Grammatical categories of english stop-words

POS	Description	Examples
IN	Preposition /Subordinate Conjunction	In , of, like, after , whether
PP	personal pronoun	I, he, it
PP\$	possessive pronoun	my, his
DT	determiner	the
CC	coordinating conjunction	and, but, or, &
TO	to	to go, to him
WP	wh-pronoun	who, what
MD	modal	could, will

Therefore, the best translation is clearly localized. Overall, the approach extracts 1866 pairs of translations, from 1675 are confirmed as correct. Thus, the accuracy of translation of the extracted Arabic relevant terms is about 89.76%.

Arabic word	Translation sequence	Frequency
خباء	tent/NN	4
خباء	notice/VVN tent/NNS	1
خباء	saw/NN	1
سوط	whip/NN	3
سوط	lash/VV	2
رمح	hand/NN spear/NN	1
رمح	spear/NN	2
سعى	chase/VVD	2
سعى	run/VVD	1

Fig. 3. Examples of translations after stop-word removal

This rate is influenced by two phenomena related to the alignment process. On the one hand, many words in the source language (respectively in the target language) are translated into multi-word expressions. This is more frequent for some arabic words which are translated as whole sentences in some cases. Second, some classical arabic words do not have english translations and are just transliterated. For example, the arabic word “الحقيقة” (which close meaning is “the sacrifice of atonement”) is replaced by “Al-aqqa”. We also give other examples in table 4 (“zakat”, “Tayammum”, “khuffs”, etc.). The NLP tools are generally unable to deal with transliterated words (labeling, lemmatization, POS tagging, etc.) as they do not exist in the arabic nor the English lexicons and thereby, can't analyze them linguistically or syntactically. Thirdly, the characteristics of Arabic as a derivational and inflectional language deteriorate the performance of the alignment phase. In addition, Arabic is characterized by a relatively free word order in a sentence or phrase. This means that the order of words in arabic sentence does not match the order of the corresponding english sentence. Finally, the translations in our corpus are semantic-based translations, meaning that instead of elaborating word-by-word translations, translators have added comments and other expressions to clarify the meaning of the original hadith. These semantic translations may have different structures and other means.

Table 4 illustrates the main results of our work by providing the top five terms and their translations for the thirteen domains.

The study of results leads us to a number of relevant observations. First and to carry out our experiments, we are not limited to the direct results of GIZA++ conducted through its implicit translation vectors, which are characterized by an ambiguity and a disability to figure out translations in some cases. Hence the idea of building vectors of translation in which we have injected the list of all possible translations of an arabic term already found in the first step of the alignment process (before elimination of stop-words) with their corresponding co-occurrences. These vectors have a large impact on the effectiveness of the alignment process, and as a consequence, on the extraction of bilingual domain terminology.

Table 4. The five top extracted couples for each domain

The beginning of the revelation بدء الوحي		Fasting الصوم		Drinks الأشربة	
هرقل	Heraclius	رمضان	Ramadan	شراب	drink
خديجة	Khadija	صوم	fast	أعال	Come here
نسب	family	جزير	jarir	كفاررة	punishment sin
ملك	king	ربّ	Lord	خمر	Alcoholic drink
اختتن	Practice circumcision	شهر	month	شحمة	fat
The sacrificed animals and hunting الذبائح والصيد		Faith الإيمان		The minor ablation الوضوء	
مغراض	Mirad	ايمان	faith	كتف	shoulder
ضراوة	hunting	زكاة	zakat	خفف	khuffs
وقنيد	beat death	قلب	heart	وضوء	ablution
خطب	Khapt	صلوة	prayer	ماء	water
وراك	pelvic	إسرائيل	Bani israel	مسح	pass
The major ablation الغسل		Sacrifices الأضحى		The dry ablation التيمم	
وضوء	ablution	محاصر	besiege	صعيد	clean earth
احلام	Wet dream	خرق	Kill body	كتف	hand
كفة	clean	ضحية	sacrifice	تيمم	perform Tayammum
طيب	look glitter scent	شياه	sheep	حائض	menses
جامع	engage in sexual intercourse	جبل	Small mountain	رمضان	Ramadan
Taraaweeh التراويح		Prayer in Seclusion الاعتكاف		Al-Aqiqah الحقيقة	
ركعة	Rakat	أدن	Pronounce	خياط	tailor
حاجي	Pour water	مكتوب	optional prayer	أمعاء	intestine
روث	dung	شيطان	Satan	معى	intestine
شفاعة	give right intercession	خفيف	upon Prophet light prayer	حشقة	Hashafa
نجس	Become impure	فلل	Up prayer	ثريد	Tharid
Foods الأطعمة					
	جنة		wheat		
	ودك		meal contain no fat		
	طعام		food		
	زمان		lifetime		
	ودك		meal contain no fat		

By conducting our experiments, we have demonstrated that it is very important to conduct a morphological analysis and to consider the lemma instead of the form of each arabic and english word during alignment and translations extraction. This allows having a good quality of these two processes. Indeed, an arabic word Wa can be aligned with an english word Wb having a form “x” in a first fragment of the text, and with the same word with a form “y” in another fragment. $Wa \rightarrow Wb(x); Wa \rightarrow Wb(y)$

To overcome this redundancy and not considering them as two separate pairs of translation, we proceed to the lemmatization. Secondly, experiments have shown in this study that the extraction of relevant terms does not depend on the size of the corpus. It achieves the best rate with the field “The beginning of the revelation” because it is semantically independent of other domains. Third, our approach is not specific to a domain and can be extended to other areas, even in modern standard Arabic. It is true that we used techniques already experienced by many researchers (such as TF-IDF), but they are not yet exploited for the Arabic language and in the field of bilingual terminology extraction, where the importance of this work.

Finally, the bilingual terminologies will be used within an AR-EN platform of terminology-based information retrieval. In fact, the latter focuses on helping the user to formulate his query and to return relevant results. It is a controlled indexing method based on the domain terminologies at this level, and to be preceded by the building of the whole bilingual ontology, which improves search results.

6 Conclusion

In this paper, we proposed and assessed an approach for bilingual domain terminology extraction; presenting the first step of bilingual ontology building; from Arabic-English corpora. To evaluate this approach, we used a vocalized version of hadith corpus. The experiments conducted in this research showed the impact of the linguistic processing and the alignment process on the extraction of terminology although we have not been able to compare against a baseline for the reasons already explained in the section 5. The results are encouraging, especially as works dedicated to Arabic language in this area are rare. However, to generalize them, we need more effort, since we used relatively conventional techniques. Yet, we believe that we have presented an attempt to bridge the gap between Arabic language and Latin languages. A comparative study is needed to better understand the impact of certain parameters such as the existence of short diacritics, the type of corpus (modern versus traditional), relevance measuring and alignment tools. As future work, we propose to consider compound terms to have a complete domain ontology as possible as we can to be used in IR. This requires the development or reuse of parsing tools. We will also try to study the impact of weighting procedures applied in the english corpus and address the issue of semantic disambiguation, since the same word may have different meanings in different areas.

References

1. Bounhas, I., Elayeb, B., Evard, F., Slimani, Y.: ArabOnto: Experimenting a new distributional approach for building arabic ontological resources. *International Journal of Metadata, Semantics and Ontologies (IJMSO)* 6(2), 81–95 (2011b)
2. Bounhas, I., Elayeb, B., Evard, F., Slimani, Y.: Organizing contextual knowledge for arabic text disambiguation and terminology extraction. *Knowledge Organization* 38(6), 473–490 (2011a)
3. El Kholy, A., Habash, N.: Orthographic and morphological processing for English-Arabic statistical machine translation. *Machine Translation* 26(1-2), 25–45 (2012)
4. Saad, M., Langlois, D., Smaïli, K.: Cross-Lingual Semantic Similarity Measure for Comparable Articles. In: 9th International Conference on Natural Language Processing, PolTAL 2014 (2014)
5. Schwenk, H., Yannick, E., Sadaf, A.: The LIUM Arabic/English statistical machine translation system for IWSLT. In: International Workshop on Spoken Language Translation, pp. 63–68 (2008)
6. Ha, L.A., Fernandez, G., Mitkov, R., Corpas Pastor, G.: Mutual bilingual terminology extraction. In: Proceedings of the 6th Conference on Language Resources and Evaluation (LREC), Marrakesh, Morocco, May 28-30, pp. 1818–1824 (2008)
7. Bouamor, D., Semmar, N., Zweigenbaum, P.: Utilisation de la similarité sémantique pour l'extraction de lexiques bilingues à partir de corpus comparables. In: Proceedings of TALN (Traitement Automatique des Langues Naturelles), Les Sables d'Olonne, France, pp. 327–338. ATALA (2013)
8. Bouamor, D., Popescu, A., Semmar, N., Zweigenbaum, P.: Building Specialized Bilingual Lexicons Using Large Scale Background Knowledge. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA, October 18-21, pp. 479–489 (2013)
9. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, pp. 1606–1611. Morgan Kaufmann Publishers Inc., San Francisco (2007)
10. Hazem, A., Morin, E.: Extraction de lexiques bilingues à partir de corpus comparables par combinaison de représentations contextuelles. In: Actes de la 20e Conférence sur le Traitement Automatique des Langues Naturelles (TALN), pp. 243–256 (2013)
11. Tamura, A., Watanabe, T., Sumita, E.: Bilingual Lexicon Extraction from Comparable Corpora Using Label Propagation. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, July 12-14 (2012)
12. Weller, M., Gojun, A., Heid, U., Daille, B., Harastani, R.: Simple methods for dealing with term variation and term alignment. In: Proceedings of the 9th International Conference on Terminology and Artificial Intelligence, Paris, France, November 8-10, pp. 87–93 (2011)
13. Simões, A., Almeida, J.: Bilingual terminology extraction based on translation patterns. *Procesamiento del Lenguaje Natural* 41, 281–288 (2008)
14. Lu, B., Tsou, B.K.: Towards bilingual term extraction in comparable patents. In: Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation (PACLIC), Hong Kong, December 3-5, pp. 755–762 (2009)

15. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, M., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses. Open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), Interactive Poster and Demonstration Sessions, Stroudsburg, PA, USA, June 23-30, pp. 177–180 (2007)
16. Eijk, P.: Automating the Acquisition of Bilingual Terminology. In: Proc. of 6th Conference of the European Chapter of the Association for Computational Linguistics, EACL 1993, pp. 113–119 (1993)
17. Sellami, R., Sadat, F., Hadrich Belguith, L.: Extraction de lexiques bilingues à partir de Wikipédia. Atelier de Traitement Automatique des Langues Africaines, JEP (conférence Journées d'Études en Parole)-TALN-RECITAL, Grenoble, France (TALAF 2012: African Language Processing) (June 2012)
18. Ayed, R., Bounhas, I., Elayeb, B., Evard, F.: Bellamine Ben Saoud N. A Possibilistic Approach for the Automatic Morphological Disambiguation of Arabic Texts. In: Proceedings of 13th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Kyoto, Japan, August 08-10, pp. 187–194. IEEE Computer Society (2012)
19. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: Proceedings of International Conference on New Methods in Language Processing, Manchester, UK (1994)
20. Salton, G., Fox, A., Wu, E., Extended, H.: boolean information retrieval. Communications of the ACM 26(11), 1022–1036 (1983)
21. Lafon, P.: Sur la variabilité de la fréquence des formes dans un corpus. Mots 1, pp. 127-165
22. Lahbib, W., Bounhas, I., Elayeb, B., Evrard, F., Slimani, Y.: A hybrid approach for Arabic semantic relation extraction. In: The 26th International Florida Artificial Intelligence Research Society (FLAIRS) Conference, St. Pete Beach, Florida, USA, May 22-24, pp. 315–320 (2013)
23. Bounhas, I., Elayeb, B., Evard, F., Slimani, Y.: Toward a computer study of the reliability of Arabic stories. Journal of American Society for Information Science and Technology 61(8), 1686–1705 (2010)
24. Harrag, F., Alothaim, A., Abanmy, A., Alomaigan, F., Alsalehi, S.: Ontology Extraction Approach for Prophetic Narration (Hadith) using Association Rules. International Journal on Islamic Applications in Computer Science And Technology 1(2), 48–57 (2013)

Towards a Configurable Database Design: A Case of Semantic Data Warehouses

Selma Khouri^{1,2} and Ladjel Bellatreche¹

¹ LIAS/ISAE-ENSMA - University of Poitiers, 86960, Futuroscope Cedex, France

{selma.khoury,bellatreche}@ensma.fr

² National High School for Computer Science (ESI), Algiers, Algeria

Abstract. Many modern software systems are designed to be highly configurable. The configuration contributes in managing evolving software and controls the cost involved in making changes to software. Several standards exist for software configuration management (IEEE 828 and IEEE 1042). Unfortunately, making database configurable did not have the same spring as for software even though it can be seen as a software product. Nowadays, we are assisting to an explosion of new deployment layouts and platforms. This situation pushed the database community to admit the slogan: “one size no longer fits all”. This motivates us to study the issue to make database design configurable. To satisfy this objective, we need to perform the following three tasks: **(i)** a deep understanding of the database design life-cycle, **(ii)** a formalization of each phase and **(iii)** an identification of the interactions between these phases. In this paper, we detail these tasks by considering the case of designing semantic data warehouses.

1 Introduction

Data are one of the most valuable resources that provide companies a serious competitive advantage. A wide range of DBMS products have been proposed in order to manage data efficiently. They are used in a large variety of new markets including conventional databases, data warehouses, scientific databases, semantic databases, social networking applications, etc. These different of databases can be summarized in one type that we call a *database product* (\mathcal{DBP}). The shareable design life-cycle of \mathcal{DBPs} includes three main phases: *conceptual design*, *logical design* and *physical design*. Its constitution has been made through a long evolution process. It starts from an *embryo* representing the physical design phase. This embryo underwent and is undergoing three main evolutions: *vertically*, *horizontally*, and *internally*. The vertical evolution adds new phases to the life-cycle. The addition of the conceptual and logical phases in the 70's to this embryo is an example of this evolution. Another example concerns the ETL phase (Extract, Transform, Load) that has been added to the life-cycle once the data warehouse technology has been launched. The *horizontal* evolution diversifies each design phase by adding new storage schemes (according to given data models), database architectures (conventional DB, *NoSQL* BD, Semantic DB,

etc.), deployment platforms (ex. Centralized, Cloud), etc. The *internal* evolution defines the steps of each design phase. We observe that the life-cycle is evolving and unfortunately most important design approaches and methodologies do not follow this evolution. Consequently, it is hard for a designer to understand and choose the alternatives offered by these evolutions, to meet her/his requirements.

A recent tendency emerging in the marketplace is to offer a *configurable storage engine*¹ (seen as ERP solution), which can be used across a broad application class, that can be tuned to the requirements of individual applications. Software configuration management² encompasses the disciplines and techniques of initiating, evaluating and controlling change to software products during and after the software engineering process. Making a configurable storage engine may be performed according two visions: (a) making the DBMS configurable or (b) making the design cycle configurable. We believe that the second option is more suitable. This vision is motivated by the citation of Michael Stonebraker who argues that “*one size no longer fits all*”³. We claim that a configurable database is feasible if the whole phases of the life-cycle are well formalized and their interactions are well captured. Our proposition in this paper is to bring us closer to this configurable engine solution. Because such a solution is very ambitious, we decided to start by focusing our study on *Semantic Data Warehouse (SDW)* systems. We propose to revisit their proposed design cycle in order to answer the two following questions: (**Q1**): what is the most relevant design choice for *SDW* systems? and (**Q2**): how can designers deploy the *SDW* system using different logical/physical representations?

We believe that answering these two questions contributes in providing a configurable engine allowing designers to choose the most appropriate design option. The proposed design cycle dedicated to *SDW* systems is composed of five design phases [4]: requirements definition, conceptual design, logical design, ETL (Extract, Transform, Load) phase and physical design. The three central phases of *DW* design cycle have been borrowed from traditional database life-cycle (conceptual, logical, physical). We claim that the cycle revisit cannot concern these phases. One of the key lessons in the DBMS field is that logical and physical data independence ensured by these abstraction levels are guarantees of good design. The revisit of the *SDW* design cycle will concern the additional phases, namely requirements definition and the ETL phase. Because existing design studies do not have a global life-cycle vision, they omit the impact of these two phases on the design process. The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 presents the design cycle formalization. Section 4 revisits the design cycle. Section 5 presents a case study using LUBM benchmark, that illustrates the revisit and proposes a design process fitting this

¹ <http://cacm.acm.org/magazines/2008/7/5376-beyond-relational-databases/fulltext>

² https://files_ifi_uzh_ch_rerg_amadeus_.../ch23-SCM-addendum.pdf

³ <http://cacm.acm.org/blogs/blog-cacm/32212-the-end-of-a-dbms-era-might-be-upon-us/fulltext>

revisit. Section 6 concludes the paper and presents some perspectives for this study.

2 \mathcal{DW} Design Methods: Single Phase Design Vision

Different design methods have been proposed in the literature covering one single design phase, or at most two design phases. They may take as inputs a collection of users' requirements and/or a set of sources. The sources considered can be structured using mono-layer DBs (usually relational sources) or multilayer DBs like *SDBs* [8]. Schemas of sources can be defined by their logical schemas [13] or conceptual schemas. Requirements can be considered as *goals* to achieve, or as business *processes* [14].

These studies may concern: the *requirements definition* phase [14], the *conceptual design* [5], the *logical design* [5], the *ETL* phase [12,2,10], or the *physical design* phase [2]. Many studies focused on the identification of multidimensional concepts.

Usually, these methods either study the design of a multidimensional \mathcal{DW} schema (conceptual, logical and/or physical) or the ETL phase. Few studies propose to design two issues conjointly (but not completely), like [11] that considers both the multidimensional design and ETL phase for designing the \mathcal{DW} (logical and physical design are ignored). In [9], authors propose an MDA-based approach for \mathcal{DW} designing schema and ETL. Different abstraction levels are defined to support the design. Some models are listed but not provided. Golfarelli et al. proposed in [6] a DW design methodology inspired by six basic principles of modern software engineering. In these last two works, the design process is not detailed.

3 Formalization of the \mathcal{DW} Design Life-Cycle

After a detailed analysis of \mathcal{DW} design literature, we propose a formalization of \mathcal{DW} cycle as follows: $\langle \mathcal{RM}, \mathcal{CM}, \mathcal{LM}, \mathcal{ETL}, \mathcal{PM} \rangle^4$ such as :

Requirements Modeling (\mathcal{RM}) : $\langle Req, Rel, Formalism \rangle$, such as: (1) *Req*: is the set of requirements collected from users and validated. (2) *Rel*: defines different types of relationships between requirements, such as *conflict*, *equivalent*, *require*, etc. (3) *Formalism*: is the formalism used for analyzing the set of requirements (like goal or process formalisms, etc.)

Conceptual Modeling (\mathcal{CM}) : our goal was to find a high level formalism that covers the most important existing static conceptual models like UML class diagram, ER models and OWL ontological models. We chose Description Logic (*DL*) formalism, which is able to capture the most popular data *class-based modeling* formalisms [3]. \mathcal{CM} is thus formally defined as follows:

⁴ Note that only model-based phases are represented, the others are process-based.

$< C, R, Ref(C), Formalism >$: (1) C : denotes *Concepts* of the model (atomic concepts and concept descriptions). (2) R : denotes *Roles* of the model. (3) $Ref : C \cup R \rightarrow (Operator, Exp(C, R))$: Ref is a function representing the various correlations between concepts. Operators can be inclusion (\sqsubseteq) or equality (\equiv). $Exp(C, R)$ is an expression over concepts and roles of \mathcal{CM} using constructors such as union, intersection, restriction, etc. (4) *Formalism* is the *formalism* followed by the global ontology model like *ER*, *RDF*, *OWL*, etc.

Logical Modeling (\mathcal{LM}) : $< \varphi_{log}(C), \varphi_{log}(R), data\ model >$: (1) $\varphi_{log}(C)$ and $\varphi_{log}(R)$ denote the logical translation of the domain objects, conforming to the underlined data model. For example, the logical translation of the concepts and the roles according to the relational data model are respectively tables and columns. (2) *data model* is the data organizational approach which can be either hierarchical, relational, object-oriented, multidimensional, etc.

The \mathcal{ETL} Modeling : is formally defined by the triplet $< G, S, M >$, such as:

- G : the global schema is defined by its conceptual, logical or physical model (as formalized in this section).
- S : Each local source S_i is a data repository, defined as follows:
 $S_i : < CM_i, I_i, Pop_i, SM_{O_i}, SM_{I_i}, Ar_i >$, with:
(1) $CM_i : < C_i, R_i, Ref_i, formalism_i >$ is the \mathcal{CM} of the source, which can be stored (as for *SDBs*) or not (\emptyset as for conventional sources). (2) I_i : presents the *instances* of the source. (3) $Pop_i : C_i \rightarrow 2^{I_i}$ is a *function* that relates each concept to its instances from the set I_i . (4) SM_{O_i} : is the *Storage Model* of CM_i . (5) SM_{I_i} : is the *Storage Model* of the instances I_i . (6) Ar_i : is the architecture of the source (mono-layer, semantic multi-layer, etc.).
- M : Mappings assertions relate a mappable element (*MapElmG*) of schema \mathcal{G} (*MapSchemaG*) to a mappable element (*MapElmS*) of schema \mathcal{S} (*MapSchemaS*), in order to run the \mathcal{ETL} process.

Physical Modeling (\mathcal{PM}) : it includes a deployment model (\mathcal{DM}) and a physical model (\mathcal{PM}).

- $\mathcal{DM} : < SS, SM_{CM}, SM_I, Ar, Deploy >$ describes how and where a system is to be deployed: (1) SS : represents the *Storage System* such as hard Disk Drive (HDD), Solid State Drive (SDD), etc. (2) SM : represents the *physical storage layout* (e.g., vertical, horizontal, binary, column store, etc.). CM subscript is for the \mathcal{CM} , I for the instances. (3) Ar : defines the database architecture (ex. conventional, semantic). (4) *Deploy*: defines the deployment platform (ex. centralized, parallel, distributed, cloud, etc.).
- $\mathcal{PM} : < \varphi_{phy}(C), \varphi_{phy}(R), I, Pop, PDS >$ such that: (1) $\varphi_{phy}(C)$ and $\varphi_{phy}(R)$ denote the physical translation of the logical objects, conforming to the deployment model. The correct notation would be $\varphi_{phy}(\varphi_{log}(C))$, but for the sake of clarity, we kept only the *phy* subscript. (2) I : represents the *instances* of the *DW*. (3) $Pop : C \rightarrow 2^I$ is a *function* that relates each *DW* concept to its instances from the set I . (4) PDS : denotes the defined optimization structures (ex. materialized views, indexes, etc.).

4 Revisiting \mathcal{DW} Design Life-Cycle

The big picture of the configurable storage engine that we aim to define is illustrated in figure 1. With the objective of realizing this engine, we propose

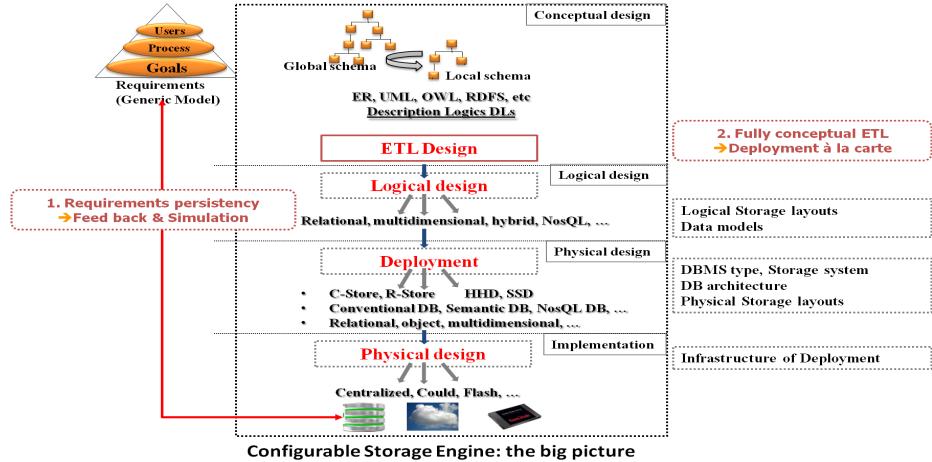


Fig. 1. Configurable storage engine

after formalizing the design life-cycle, to revisit \mathcal{DW} design by highlighting two interactions : (1) Requirements interactions and (2) The ETL design phase interactions.

4.1 Requirements Persistence

In order to answer question (Q1), it would be efficient to simulate the \mathcal{DW} behavior on the different design alternatives. When analyzing the design cycle, we identified that users' requirements represent the only resource for designing \mathcal{DW} systems, and consequently for predicting their behavior à priori. We propose as a first contribution to persist users' requirements in \mathcal{DW} system in order to use them all along the \mathcal{DW} life cycle. Requirements are divided in two sets: *data* requirements and *process* requirements. They allow the identification of: (1) the set of relevant concepts used by the target system, forming a *dictionary*, which is materialized by the \mathcal{CM} , then \mathcal{LM} and \mathcal{PM} and (2) the set of treatments that the system should answer. Conventional DW design keeps trace of the data requirements part and ignores the process requirements part. The revisited design cycle aims to save both parts in order to predict the DW behavior or to evolve. In order to keep trace of this design process, we propose to extend the proposed cycle formalization by function ($\mathcal{LReq} : Req \rightarrow 2^{CU2^RU2^{Ref}}$) that links each requirement to the concepts and properties it defines. The connection is thus formalized as follows: $\langle \mathcal{RM}, \mathcal{CM}, \mathcal{LReq} \rangle$, and will be stored in the

final \mathcal{DW} system (as illustrated in the case study). The following interactions (of \mathcal{LM} and \mathcal{PM}) are already materialized in the default cycle.

4.2 Conceptual ETL

In order to answer question (Q2), we need to provide a design process where the \mathcal{DW} can be deployed "à la carte". We identified that the main limitation that constraints this goal is the definition of the mappings (between the global and sources schemas) at the logical level. We proposed as a second contribution to define the ETL phase entirely at the conceptual level, in order to free this phase from all implementation constraints. The ETL formalization is consequently instantiated as follows:

- (1) G is defined by its $\mathcal{CM} : < C, R, Ref(C), Formalism >$.
- (2) Each source S_i is defined as follows: $< CM_i, I_i, Pop_i, SM_{O_i}, SM_{I_i}, Ar_i >$. To ensure a conceptual ETL design, CM_i must be available. It is explicitly defined (as in $SDBs$) or defined by a reverse engineering process (as for conventional sources).
- (3) Mappings are defined between concepts, roles or expressions of \mathcal{CM} and \mathcal{CM}_i

The final \mathcal{DW} obtained is formalized as follows:

$< CM_{DW}, I_{DW}, Pop_{DW}, SM_{CM_{DW}}, SM_{I_i}, Ar_i >$ extended by the requirements part, which gives the following formalization:

$< RM_{DW}, CM_{DW}, \mathcal{LReq}, I_{DW}, Pop_{DW}, SM_{CM_{DW}}, SM_{I_i}, Ar_i >$.

- $RM_{DW} : < Req, Rel, Formalism >$, where requirements Req are collected from users.
- CM_{DW} is defined from G (CM_{DW} is equivalent to G or a fragment of G). Three scenarios are possible: (1) $CM_{DW} = G$: G corresponds exactly to user's requirements; (2) $CM_{DW} \subseteq G$: CM_{DW} is extracted from G using modularity methods; (3) $CM_{DW} \supseteq G$: G does not fulfill the whole users' requirements. The designer extracts the fragment of the G corresponding to requirements and enriches it with new concepts and properties.
- I_{DW} and Pop_{DW} functions are obtained by the ETL process, which aliment each class by its corresponding instances according to the defined mappings. Some studies defined conceptual ETL processes (algorithms) like [2,12].
- SM_{CM}, SM_I , and Ar are *configurable* and defined by the designer.

5 A Case Study: Towards a Configurable Design Process

We instantiate the design framework proposed to define a semantic \mathcal{DW} that takes as inputs a set of semantic databases ($SDBs$) as sources, referencing domain ontology supposed existent, and a set users goals (requirements). We chose $SDBs$ because they are perfect candidates for a fully conceptual ETL. $SDBs$ present sources that store their conceptual model in the form of a local ontology. The mappings can thus be defined between these ontologies and the domain ontology. We chose for this scenario Oracle $SDBs$. Oracle uses *OWLPrime*⁵ (a

⁵ <http://www.w3.org/2007/OWL/wiki/OracleOWLPrime>

subclass of *DL*). The adopted scenario consists in creating 6 Oracle *SDBs*, where each source references *Lehigh University BenchMark* ontology (LUBM)⁶ considered as the domain ontology. The sources are populated using UBA, a Data Generator tool provided by LUBM. Different complex mappings are defined between local ontologies of sources and LUBM ontology. We used requirements provided by LUBM that we adapted as business requirements.

For the first design phase, we opted for a goal oriented requirement model (proposed in [7]) as usually used in *DW* systems. The goal model instantiates the *RM* as follows: <*Requirements*: set of goals, *Relationships*: (And, Or, Influence), *Formalism*: Goal Model>. Function *LReq* can be easily implemented using an Ontology editor like Protege. This connection informs the designer about the most relevant data to store in the *DW* model. A *DW* ontology (*DWO*) (the warehouse \mathcal{CM}) is defined from the domain ontology *DO* by extracting all concepts and properties used by user goals using a modularity method. We use the ETL algorithm we proposed in [1] order to populate *DWO* classes by instances of *SDBs* classes. Each ETL operator is translated to an ontological query using Sparql language. *DW* logical schema is generated by translating the *DWO* populated with instances, to a relational model. Three main approaches can be used to translate OWL ontology to a relational schema: *vertical*, *binary* and *horizontal*, which represents a logical deployment 'la carte'. We propose the use of a *SDB* (eg. Oracle) for the implementation of the final *DW*. The ontology part is already stored in the *SDB*. It is achieved using Oracle SQL Loader. Oracle ontology meta-schema is extended by the goal model. The set of requirements can thus be retrieved at any moment using SQL or Sparql queries. Finally, this case study scenario is implemented in a case tool implementing the proposed configurable design process, instantiated for semantic *DW* design.

6 Conclusion

DBPs are at the heart of many systems and applications in almost all organizations today. Designing a *DBP* is a complex task which is supported by a design life-cycle. This cycle is continuously evolving in order to meet new requirements, which greatly diversifies the design options. This diversity may be handled by Configurability, a concept borrowed from software management. A configurable design process must offer the possibility of designing *DBPs* using different design alternatives. We propose in this paper to revisit the design life-cycle in order to provide a configurable design process dedicated for *DW* systems. Two main contributions are proposed during this revisit: a persistent view of requirements in the *DW* and their relationships with the following phases, and an ETL design defined at the conceptual level. These contributions were preceded by a formalization of the design cycle phases. A case study is given illustrating the proposal. This study opens various perspectives, among them: the definition of a complete configurable design method supported by a simulation tool, that uses benchmarks and cost models in order to simulate the *DBP* behavior during all

⁶ [http://www.lehigh.edu/\\$\sim\\$zhp2/2004/0401/univ-bench.owl](http://www.lehigh.edu/\simzhp2/2004/0401/univ-bench.owl)

its life-cycle, and that can be adapted to support new design requirements. This proposition would include the formalization of the whole \mathcal{DBP} cycle including additional phases like optimization, tuning and personalization. Finally, it is important to evaluate the evolution of requirements on the \mathcal{DBP} system.

References

1. Bellatreche, L., Khouri, S., Berkani, N.: Semantic data warehouse design: From ETL to deployment à la carte. In: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (eds.) DASFAA 2013, Part II. LNCS, vol. 7826, pp. 64–83. Springer, Heidelberg (2013)
2. Berkani, N., Bellatreche, L., Khouri, S.: Towards a conceptualization of etl and physical storage of semantic data warehouses as a service. Cluster Computing 16(4), 915–931 (2013)
3. Calvanese, D., Lenzerini, M., Nardi, D.: Description logics for conceptual data modeling. In: Logics for Databases and Information Systems, pp. 229–263 (1998)
4. Golfarelli, M.: From user requirements to conceptual design in data warehouse design a survey. In: Data Warehousing Design and Advanced Engineering Applications Methods for Complex Construction, pp. 1–16 (2010)
5. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: a conceptual model for data warehouses. International Journal of Cooperative Information Systems 7(02n03), 215–247 (1998)
6. Golfarelli, M., Rizzi, S., Turricchia, E.: Modern software engineering methodologies meet data warehouse design: 4WD. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 66–79. Springer, Heidelberg (2011)
7. Khouri, S., Bellatreche, L., Boukhari, I., Bouarar, S.: More investment in conceptual designers: Think about it? In: CSE 2012, pp. 88–93 (2012)
8. Khouri, S., Boukhari, I., Bellatreche, L., Jean, S., Sardet, E., Baron, M.: Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. Computers in Industry, 799–812 (2012)
9. Mazón, J.-N., Trujillo, J.: An mda approach for the development of data warehouses. Decision Support Systems 45(1), 41–58 (2008)
10. Nebot, V., Berlanga, R.: Building data warehouses with semantic web data. Decision Support Systems (2011)
11. Romero, O., Simitsis, A., Abelló, A.: *GEM*: Requirement-driven generation of ETL and multidimensional conceptual designs. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 80–95. Springer, Heidelberg (2011)
12. Skoutas, D., Simitsis, A.: Ontology-based conceptual design of etl processes for both structured and semi-structured data. International Journal on Semantic Web and Information Systems (IJSWIS) 3(4), 1–24 (2007)
13. Song, I.Y., Khare, R., Dai, B.: Samstar: a semi-automated lexical method for generating star schemas from an entity-relationship diagram. In: Proceedings of the ACM Tenth International Workshop on Data Warehousing and sLAP, pp. 9–16. ACM (2007)
14. Winter, R., Strauch, B.: A method for demand-driven information requirements analysis in data warehousing projects. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, p. 9–19. IEEE (2003)

Describing Research Data: A Case Study for Archaeology

Nicola Aloia¹, Christos Papatheodorou^{2,3}, Dimitris Gavrilis³, Franca Debole¹,
and Carlo Meghini¹

¹ Istituto di Scienza e Tecnologie dell'Informazione, National Research Council, Pisa,
Italy

{Nicola.Aloia,Franca.Debole,Carlo.Meghini}@isti.cnr.it

² Dept. of Archives, Library Science and Museology, Ionian University, Corfu, Greece

³ Digital Curation Unit, Institute for the Management of Information Systems,

'Athena' Research Centre, Athens, Greece

{c.papatheodorou,d.gavrilis}@dcu.gr

Abstract. The growth of the digital resources produced by the research activities demand the development of e-Infrastructures in which researchers can access remote facilities, select and re-use huge volumes of data and services, run complex experimental processes and share results. Data registries aim to describe uniformly the data of e-Infrastructures contributing to the re-usability and interoperability of big scientific data. However the current situation requires the development of powerful resource integration mechanisms that step beyond the principles guaranteed by the data registries standards. This paper proposes a conceptual model for describing data resources and services and extends the existing specifications for the development of data registries. The model has been implemented in the context of the ARIADNE project, a EU funded project that focuses on the integration of Archaeological digital resources all over the Europe.

Keywords: Data registries, Research infrastructures, Interoperability, Archaeological digital resource.

1 Introduction

Extremely large scientific datasets are being generated and the issues for identifying, locating, re-using and exploiting data are getting more difficult and imperative. This data deluge affects the way research is carried out leading to a data-oriented paradigm. Data integration functionalities, data analysis, data mining and visualization tools should support this shift of the research and scholar communication paradigm. For this purpose Global Research Data Infrastructures infrastructures are being developed to assure the interoperability and discoverability of scientific resources and cope with the (i) structural (syntactic) heterogeneity of data organized in datasets, following particular database schemas, or in collections described by different metadata schemas at collection

level as well as at item level, (ii) semantic heterogeneity caused by the diversity of vocabularies used for the description of artefacts, activities, events, temporal periods, workflows and geospatial data, (iii) diversity of metadata schemas that requires their semantic integration through mappings to upper level conceptual schemas [1,2].

This paper is inspired by the work done on ARIADNE project¹, an EU funded project that develops a data registry aiming to the integration of archaeological research data. Due to the huge volume of data the main challenge is to provide an environment for establishing the degree of relatedness between data resources, in order to plan their integration. The achievement of such a goal needs the design of a system based on (i) algorithms for estimating the degree of relatedness between any two data resources and (ii) a Catalog of the existing data resources for tuning and executing the algorithms. The paper presents the conceptual model of the Catalog, named ARIADNE *Catalog Data Model* (ACDM) that steps beyond accessibility and re-usability requirements and extends the existing data registry standards².

2 Background

The main processes for making data understandable and shareable are standardization and registration. ISO/IEC 11179 facilitates acquisition, registration, reuse, interchange and sharing of data [3]. Based on this standard significant efforts have been made to develop metadata registries [4,5,6,7]. The most recent effort is the Open Metadata Registry³, formerly named the National Science Digital Library (NSDL) Registry, which hosts vocabularies and their terms (concepts), metadata schemas and their elements and details about the agents (persons or corporate bodies) who have added content to the registry. Its implementation was based on the W3C standard Simple Knowledge Organization System (SKOS) and hence the user can search for vocabularies, terms, metadata schemas and metadata elements and retrieve their descriptions via a SPARQL endpoint.

DCAT⁴ is an RDF vocabulary, recently published by the Government Linked Data Working Group at W3C as a recommendation to describe datasets and catalogs on the Web in order to enable their discoverability and consumption by services. The DCAT model “*is well-suited to representing government data catalogues such as Data.gov and data.gov.uk*” and has been proposed as a tool for publishing datasets as Open Data [8]. Currently various datasets have been published according to the DCAT specifications and various European projects officially recommend its adoption. DCAT brings a number of classes from other well known vocabularies such as foaf:Agent, skos:Concept, as well as a set of relations among them. The main classes of the model are *dcat:Catalog* that represents a curated collection of metadata about datasets, *dcat:Dataset* that represents a

¹ <http://www.ariadne-infrastructure.eu/>

² The system is available at: <http://schemas.cloud.dcu.gr/ariadne-registry/>

³ <http://metadataregistry.org/>

⁴ <http://www.w3.org/TR/vocab-dcat/>

published and curated collection of data and *dcat:Distribution* that represents each dataset might be available in different formats or different endpoints.

3 ARIADNE Catalog Data Model

The proposed Catalog will support the browsing and querying of data resources and services providing useful information on each of them. We estimate its size to reach the order of thousands resources and therefore we are going to design tools for the exploration of this space and the *discovery* of archaeological resources. We plan to offer two kinds of discovery: a *semantic discovery*, allowing user to identify the resources, that relate to a specific topic, event or spatio-temporal region; a *similarity discovery*, allowing user to provide (the identifier of) a data resource as input and to obtain in return (descriptors of) the resources that are similar to the given one, ranked in decreasing degree of similarity.

In this section we describe the ARIADNE *Catalog Data Model* (ACDM), defined to register information about resources that are scattered amongst different collections, inaccessible and unpublished fieldwork reports “grey literature” and in publications. These resources come in three different types: *Data Resources*, including the resources that are containers of data such as databases and collections; *Language Resources*, including the resources related to the formal languages used in Data Resources, such as vocabularies, metadata schemas and mappings; and *Services*, including the resources offering some kind of functionality in the archaeological domain.

We built our model around the DCAT vocabulary, which we expanded by adding classes and properties that were needed for best describing the ARIADNE assets. Its adoption places ARIADNE in an ideal position for publishing archaeological data resources as Open Data. As illustrated in Figure 1 the central notion of the model is the class *ArchaeologicalResource*, specialized in:

- *DataResource*, whose instances represent the various types of data containers owned by the ARIADNE partners and lent to the project for integration. This class is created for the sole purpose of defining the domain and the range of a number of associations. It is therefore an abstract class, whose instances are inherited from sub-classes.
- *LanguageResource*, having as instances vocabularies, metadata schemas, gazetteers and mappings (between language resources). As new resources of linguistic nature are added to the Catalog (such as subject heading systems and thesauri) the corresponding classes will be added to the model as sub-classes of this class. To describe language resources we have used ISO/IEC 11179 “Specification and Standardization of Data Elements” [3].
- *Services*, whose instances represent the services owned by the Ariadne partners and lent to the project for integration.

Classes with a more auxiliary role are: *DataFormat* whose instances represent the formats that realize metadata schemas, or structure the records of datasets; *DBSchema* to represent the instances of database schemas.

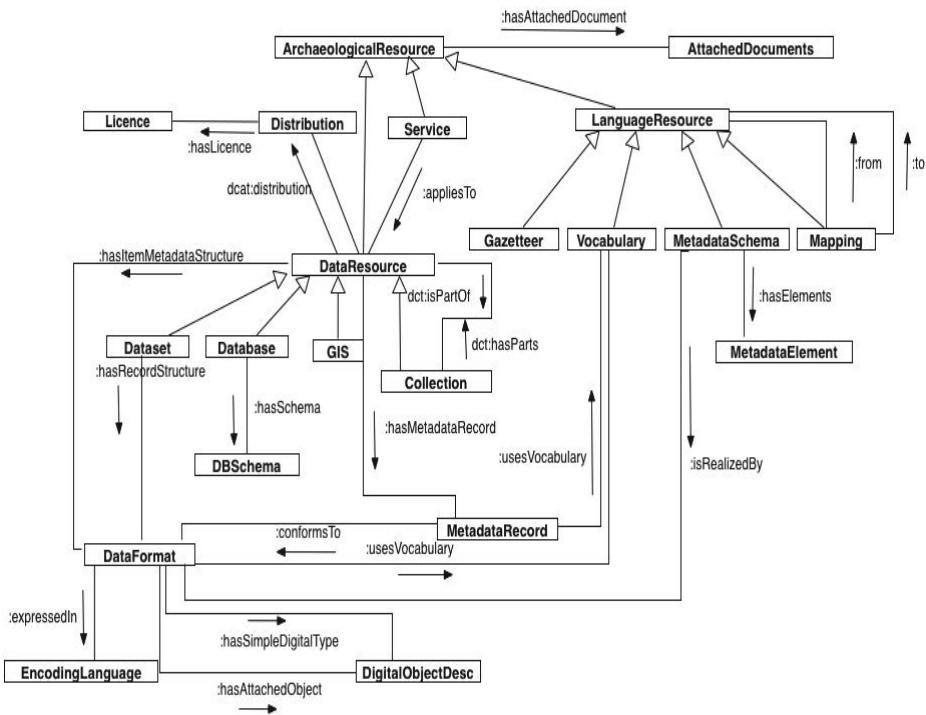


Fig. 1. The ACDM as a UML diagram

3.1 ArchaeologicalResource

The *ArchaeologicalResource* class defines the properties common to its subclasses, mostly using the terms of the DCAT vocabulary, to which it adds properties for specifying: the access policy and the original identifier of the resource. The main associations having this class as domain are:

- **dct:isPartOf** associates any archaeological resource in the catalog with that catalog.
- **dct:publisher**: associates any archaeological resource with an agent responsible for making the resource publicly available.
- **dct:creator**: associates any archaeological resource with an agent primarily responsible for creating the resource.
- **owner**: associates any archaeological resource with an agent that is the legal owner of the resource.
- **legalResponsible**: associates any archaeological resource with a person holding the legal responsibility of the resource.
- **scientificResponsible**: associates any archaeological resource with a person holding the scientific responsibility of the resource.

- **technicalResponsible**: associates any archaeological resource with a person holding the technical responsibility of the resource and contact person.
- **ariadneSubject** associates any archaeological resource with one or more archaeological subjects defined by ARIADNE, namely: Fieldwork databases, Event/intervention databases, Sites and monuments databases, Scientific databases, Artefacts, Burials.
- **dct:subject** associates any archaeological resource with a subject drawn from an existing vocabulary.

3.2 DataResource

This class specializes the class *ArchaeologicalResource*, and has as instances the archaeological resources such as databases, GIS, collections or datasets. Two important attributes of this class are **dct:temporal** and **dct:spatial**, giving the spatial and temporal coverage of each instance data resource. The attributes will be used for establishing the degree to which two data resources are worth integrating. The main associations having this class as domain are:

- **dct:isPartOf** associates a data resource with the collections which the data resource is part of.
- **dcat:distribution**: associates a data resource with the distributions, *i.e.* the accessible forms of the resource.
- **hasItemMetadataStructure**: associates a data resource with the format of the metadata of the members (or items) of the data resource (*e.g.* metadata of each record in a dataset, or of each item in a collection).
- **hasMetadataRecord**: associates a data resource with the metadata of the resource as created by the organization holding the resource (for instance, the record describing a dataset in the organization holding the dataset).

The class has the following subclasses:

Collection: This class has as instances collections in the archaeological domain. The items in a collection are data resources themselves; for instance, a collection may include a textual document, a set of images, one or more datasets and other collections. For interoperability, Collection is a sub-class of *dcmitype:Collection*. The main association having this class as domain is **dct:hasParts**, which associates a collection with the data resources that are in the collection. This association is used in the ARIADNE Catalog only for stating membership of data resources in collections, since the Catalog does not store information on individual objects.

Database: This class has as instances databases, defined as a set of homogeneously structured records managed through a Database Management System (such as MySQL), recorded as an attribute of the class. The main association having this class as domain is **hasSchema**, which associates a database with the schema defining the structure of the data in the database. Such schema is an instance of the class DBSchema.

Dataset: This class is a specialization of the classes *DataResource* and *dcat:Dataset*, and it has archaeological datasets as instances. An archaeological

dataset is defined as a set of homogeneously structured records that are not managed through a Database Management System. The main association having this class as domain is **hasRecordStructure**, which associates a dataset with a data format defining the structure of its records. Such format is an instance of the class **DataFormat**.

GIS: This class is a specialization of the class *DataSource*, and has as instances Geographical Information Systems (GISs). The GIS technology used for each instance is modelled as an attribute of the class.

3.3 LanguageResource

This is the class of all language resources described in the Catalog for the purposes of re-use or integration within the ARIADNE community. A language resource is a resource of a linguistic nature, whether in natural language (such as a gazetteer) or in a formal language (such as a vocabulary or a metadata schema). It also includes mappings, understood as associations between expressions of two language resources that may be of a formal (e.g., sub-class or sub-property links) or an informal (e.g., natural language rules) nature. The **LanguageResource** have as instances vocabularies, metadata schemas, gazetteers and mappings (between language resources). The most significant subclasses of the class are:

MetadataSchema: This subclass has as instances metadata schemas used in the archaeological domain.

Vocabulary: This is a subclass has as instances vocabularies used in the archaeological domain.

Mapping: An instance of this class represents a mapping between two language resources (e.g. metadata schemas).

3.4 Service

The modelling of the services to be integrated by ARIADNE is at a preliminary stage of development. The goal is to provide the primitives for describing the services developed by the project partners for which integration or reuse can be envisaged. A preliminary survey has brought about the following categories: services that make use of GIS software; services that make use of databases management systems; *ad hoc* systems developed in-house that do not use any of the previous technologies; composite services that use a combination of the previous categories. An ARIADNE service is therefore understood as an instance of one of the four categories of software listed above.

Another important feature of services to be considered in the context of ARIADNE is how they can be accessed. From the preliminary investigation, the following types can be distinguished: services to be used locally, requiring installation on a specific hardware/software platform; services to be used locally independent from any specific hardware/software platform; services to be used as web applications; web services based on a standard protocol. For the first three categories, it is important to know whether they provide Application Programming Interfaces (API) and whether they are Open Source (Figure 2).

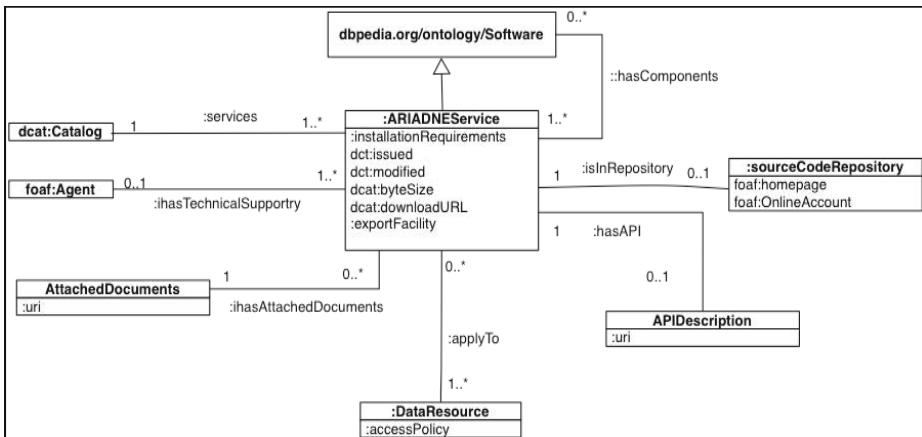


Fig. 2. The Ariadne Services Data Model

A third feature, relevant to the service description in the ARIADNE context is the kind of functionality offered by the services (e.g. map viewer, data entry system, etc.). We did not find a shared ontology to express the characteristics of services as discussed above. The best approximation that we found is the ontology adopted in DBpedia⁵ for describing software, so we defined the `ARIADNEService` class as a specialization of the `DBpedia-Software` class. The main associations having this class as domain are:

- **applyTo:** the `DataResource` to which the service can be applied.
- **isInRepository:** if the source code is available in a repository URI and other information like credential to access the repository are supplied.
- **hasAttachedDocuments:** the documents that are attached to a service for illustration purposes.
- **hasTechnicalSupport:** the person responsible for the technical support.
- **hasAPI:** if the service provide an API, a description must be supplied.
- **hasComponents:** a service may include some other components.

4 Conclusions

The main goal of the ARIADNE project is to “*to integrate the existing archaeological research data infrastructures so that researchers can use the various distributed datasets and new and powerful technologies as an integral component of the archaeological research methodology*”. In order to achieve this goal, it is necessary to (i) gather information about the existing data resources and services in the archaeological domain, and (ii) to implement advanced search functionalities

⁵ <http://dbpedia.org/ontology/Software>

on this information in order to support the discovery of resources that make good candidates for integration. As a necessary step towards the realization of the first objective, we have set out to design a data model for representing archaeological resources. In the interest of understandability, usability and interoperability, the data model is an extension of the DCAT W3C Recommendation and includes classes and properties from other well-known vocabularies, such as Dublin Core, DBpedia and FOAF. As a necessary step towards the realization of the second objective above, we have implemented functionality for the persistence and the population of the Catalog.

Much work lies ahead. First, we aim at making the content of the ARIADNE Catalog available as Linked Data. Most importantly, we aim at making both the Catalog and the services built around it, available to the rest of the archaeological community, by deploying both on the web. This is by far the most ambitious of our goals, and it requires a thorough effort, for completing the on-going implementations, validating the results, documenting the services so that users outside ARIADNE can learn them with minimum effort, and strengthening the system so that it can sustain usage at a global level.

Acknowledgment. This work has been partially funded by the European Commission under ARIADNE funded in the theme Research infrastructures (Grant agreement no: 313193).

References

1. Castelli, D., Manghi, P., Thanos, C.: A Vision Towards Scientific Communication Infrastructures: On Bridging the Realms of Research Digital Libraries and Scientific Data Center. *International Journal on Digital Libraries* 13(3-4), 155–169 (2013)
2. Thanos, C., Manegold, S., Kersten, M.: Big Data – Introduction to the Special Theme. *ERCIM News* (89), 10–11 (2012)
3. ISO 11179 Part1 Framework for the Specification and Standardization of Data Elements (2004)
4. Caplan, P.: Metadata Fundamentals for All Librarians, American Library Association (2003) ISBN 9780838908471
5. DCMI Registry, <http://dublincore.org/dcregistry/>
6. Jeong, D., Baik, D.K., Park, S.H.: A Practical Approach: Localization-Based Global Metadata Registry for Progressive Data Integration. *J. Info. Know. Mgmt.* 2, 391–401 (2003)
7. Heery, R., Gardner, T., Day, M., Patel, M.: DESIRE metadata registry framework, Deliverable 3.5. DESIRE II - Development of a European Service for Information on Research and Education II (1999), <http://web.archive.org/web/20080513183558/> (retrieved)
8. Goedertier, S.: DCAT application profile for data portals in Europe. (2013), https://joinup.ec.europa.eu/asset/dcat_application_profile/ (retrieved)

Enriching Semantically Web Service Descriptions

Maricela Bravo¹, José Rodríguez², and Alejandro Reyes¹

¹Systems Department, Autonomous Metropolitan University
Azcapotzalco, DF, CP 02200 - Mexico

{mcbr, jaro}@correo.azc.uam.mx

²Computing Department, CINVESTAV-IPN
Gustavo A. Madero, DF, CP 07300 - Mexico
rodriguez@cs.cinvestav.mx

Abstract. Service Oriented Computing (SOC) has incrementally been adopted as the preferred programming paradigm for the development, integration and interoperation of large and complex information systems. However, despite its increasing popularity, the SOC has not achieved its full potential yet. This is mainly due to the lack of supporting tools to enrich and represent semantically Web service descriptions. This paper describes a solution approach for the automatic representation of Web service descriptions and their further semantic enrichment between operation names based on the calculation of four semantic similarity measures. The enrichment approach is accurate because the final decision is done through a voting scheme, in the case of inconsistent results, these are not asserted into the ontology. Experimentation shows that although few similarity relationships are found and asserted, they represent an important step towards the automatic discovery of information that was previously unknown.

Keywords: Public Web Service Descriptions, Ontology Representation, Semantic Web Services.

1 Introduction

Service Oriented Computing (SOC) has been adopted as the preferred programming paradigm for the development of complex information systems. This trend has led to the emergence of service repositories, service frameworks and many supporting technologies which offer facilities for searching, discovering, selecting and invoking Web service operations. However, the SOC has not achieved its full potentiality, mainly because search and invocation of Web service operations still lacks of the level of automation and that facilitates that any service requestor exploits any public available Web service.

Whenever a service requestor searches for a service in public repositories, he obtains a list of services that syntactically match the keywords provided, then he has to check one by one in order to identify which of these services satisfies his *functional* requirements. This is not an easy task as the majority of available Web services are described in WSDL 1.0 1.1, or WSDL 2.0. WSDL offers syntactical information

regarding the service address, name, operations, input and output messages and all the required information to invoke the service. However, WSDL lacks of functional information regarding the use of operations and parameter values. To advance on a feasible solution, it is necessary to build repositories of semantically enriched web services; but these repositories must reuse existing services. The work reported in this paper represents a step towards this end.

Semantic Web Services, introduced by McIlraith et al. in [1], rely on the incorporation of ontologies to enhance service descriptions. According with T. Gruber [2] "An ontology is an explicit specification of a conceptualization". An ontology also defines formally the relationships that exist between terms and a set of axioms which detail and restrict the concepts. Inspired by the concept of Semantic Web Services, in this paper we introduce a solution for the automatic representation of Web service descriptions as ontological models and their further semantic enrichment based on semantic similarity measures. The type of semantic relationships that are discovered between ***operation names*** are "*isSimilarTo*" and "*isDifferentTo*".

Continuing with the work reported in [3], in this work *semantic enrichment of Web services* is the improvement of Web service descriptions by means of an ontological representation. This process consists of three general phases: 1) *Web Service Ontology Generation*, this phase consists of parsing Web service descriptions and applying a predefined ontology template to automatically produce its corresponding ontology model; 2) *Discovery of Semantic Relationships*, which consists of calculating similarity measures between operation names and for each set of results calculate the upper and lower thresholds, which are then used to identify semantic relationships: *isSimilarTo*, *isDifferentTo* and *unDefined*. To decide on semantic relationships a voting schema is used. An important design goal for this phase was to build a module capable of incorporating ***any set of similarity measures***. 3) *Instantiate new Semantic Relationships*, which consists of asserting semantic relationships between service operations into the ontology.

The rest of the paper is organized as follows: in Section 2, related work is presented; in Section 3, the discovery of semantic relationships between service operations is detailed; in Section 4, the ontological model for the representation of Web services is presented; in Section 5, the experimental setup is described; in Section 6, experimental results are evaluated; and finally conclusions are presented in Section 7.

2 Related Work

In this section related work concerning semantic Web services, service directories and similarity measures is described. McIlraith et al. [1] described an approach to markup Web services to enable automatic Web service discovery, execution, composition and interoperation. Sycara et al. [4] presented one of the first semantic languages and infrastructure to mark up Web services: DAML-S. An ontology mapping solution was presented by Pathak et al. [5] to support the translation of service ontologies and user ontologies facilitating service discovery and matchmaking using non functional

characteristics. Klush, Fries and Sycara [6] presented OWLS-MX a OWL-S service matchmaker which incorporates reasoning on logically defined preconditions and effects. The main limitation with OWL-S is that the majority of public available service descriptions are in WSDL language, few OWL-S public service descriptions exist. Gomadam et al. [7] introduced the notion of semantic template to capture the requirements of a service requestor using SAWSDL and model references. Authors also describe an automatic approach for Web service composition addressing the problems of process heterogeneities and data heterogeneities. Du, Song and Munro [8] described a method for transforming existing Web service descriptions into an enhanced semantic Web service framework which incorporates composition relationships between services. Their composition relationship definition links a service output with a different service input through a similarity measure. However, this output-input relationship rather defines a data type compatibility than a functional compatibility. Elgazzar, Hassan and Martin [9] presented an approach to improve Web service discovery by clustering Web services into functionally similar groups. The main limitation of their approach is that they do not provide any semantic representation of clusters or mechanisms to infer and reason about their results.

OWL-S is an ontology-based service description language [15], which supplies service providers with a set of constructs for describing the properties and capabilities of their Web services. An OWL-S Service presents a service Profile, is described by a Service Model (or Process Model); and supports a Service Grounding. The Semantic Annotation for WSDL (SAWSDL) specification [16] defines a set of mechanisms to add annotations to WSDL documents, such annotations reference ontologies. SAWSDL is helpful for the discovery and invocation of Web services. The Web Service Modeling Ontology (WSMO) is a complete ontological model [17] that describes: Ontologies, Web Services, Goals, and Mediators. The WSMO incorporates the Web Service Modeling Language (WSML), a language for the specification of Semantic Web services.

Related works rely on the incorporation of ontologies as a mechanism to achieve semantic interoperability. The main drawbacks of these related works is that users need to provide their ontology concepts and create manually mappings between ontologies. None of reported works have presented a fully automated enrichment approach using public available Web services described with different WSDL versions.

3 Discovering Semantic Relationships

The objective of this phase is to find similarities or differences between operation names using semantic measures and establish new semantic relationships between individuals into the ontology.

Calculate Semantic Similarities between all Operations. The operation names are short texts from one to seven words. These names are written in several formats and contain no relevant information in some cases. So, in order to get the similarity between operations a pre-processing phase is required. Preprocessing phase involves

obtaining lexical units that are part of the operation name. First, text normalization is performed in order to transform operations names into a single canonical form, for example: *getFlightPrice*. Then, lexical units are obtained from operation names, for example: [*get*][*Flight*][*Price*]. Finally, the processing also includes a lexical discrimination of several words that do not contain important meaning in the operations, which area: *http, for, return, result, soap*. These words are filtered out prior to calculate semantic similarity between operations.

The level of similarity between operations pairs is performed by calculating the average of semantic similarity measures between all words filtered. Four measures were used for this process: the Wu and Palmer measure [10] that calculates semantic similarity by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of the lowest common subsumer; the Lin measure [11], which is a universal definition of similarity in terms of information theory that is not directly stated as in earlier definitions, rather, it is derived from a set of assumptions; the path measure [12] relies on the length of the shortest path between two synsets for their measure of similarity limiting to IS-A links and scale the path length by the overall depth of the taxonomy; the Lesk measure [13] proposed that the relatedness of two words is proportional to the extent of overlaps of their dictionary definitions.

Calculate Mean and Standard Deviation. We calculate the arithmetic mean and standard deviation using the similarities of all operation pairs. Arithmetic mean is obtained from Equation 1, while standard deviation is defined from Equation 2.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (1)$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (2)$$

Suppose we have a data set, x_1, \dots, x_n then

N is the number of n values of our data set

x_i is a data contained in our data set

We use the arithmetic mean and standard deviation to obtain the upper and lower thresholds. The upper threshold is used to identify those operation pairs that have a positive similarity according with the semantic measure applied. The lower threshold is used in the identification of operation pairs that are definitively different in accordance with the semantic measure applied.

Set the Upper and Lower Thresholds and Discover Relationships. Two thresholds are defined from the arithmetic mean and standard deviation in order to determine the limits that represent the corresponding semantic relationships. The upper threshold defines the limit for the similarity relationship *Operation_i isSimilarTo Operation_j*.

$$T_{upper}(sim) = \bar{x} + \sigma \quad (3)$$

where \bar{x} is the arithmetic mean and σ is the standard deviation.

The lower threshold defines the limit for the difference relationship between operations $Operation_i$ *isDifferentTo* $Operation_j$.

$$T_{lower}(sim) = \bar{x} + \sigma \quad (4)$$

where \bar{x} is the arithmetic mean and σ is the standard deviation.

For each similarity between operations that are over the upper threshold a semantic relationship *isSimilarTo* is defined. Also, for each similarity under the lower threshold a semantic relationship *isDifferentTo* is defined.

For those operation pairs whose calculations resulted under the upper threshold and over the lower threshold no semantic relationship is established, because there is no numerical certainty to establish the similarity or difference.

4 Ontological Model

The general model was designed to represent the following service description implementations: WSDL 1.0, WSDL 1.1 and WSDL 2.0. It consists of: a *Service class*, which represents a WSDL service description; *Endpoint class*, which specifies a unique network address that the service consumer uses to invoke the methods of the service; *Binding class*, which specifies the SOAP binding style and transport; *Interface class*, the *portType* of a Web service defines the operations that can be invoked, and the input and output messages that are used to execute the operation, in the ontological model depicted in Figure 1, instead of a *portType* class, an *Interface class* was created to make this model compatible with the WSDL 2.0 specification; *Operation class*, represents the methods offered by the service interface, the WSDL specification defines an input message and output message for each operation; *Parameter class*, the *Parameter class* represents the super class of the *ParameterInput class* and the *ParameterOutput class*.

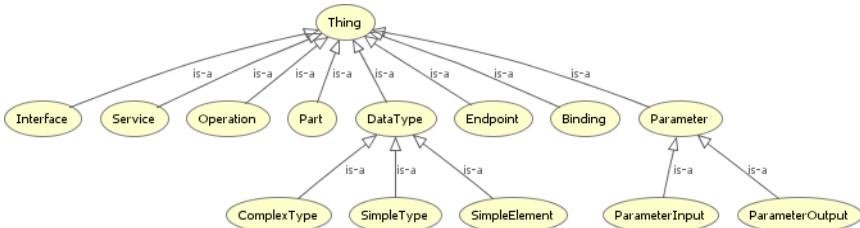


Fig. 1. General ontological model for the representation of Web services

5 Experimentation

An experiment was executed using the test collection OWLS-TC3, which contains 1080 Web service descriptions. A subset of 43 Web services was selected using as a criteria their file names - file name starting with "Country" - this selection was done arbitrarily.

1) *Web Service Ontology Generation.* SDWS was executed following a sequence of predefined steps: selection of any set of Web services; uploading the set of selected services, parsing the services according to their representation language; populating a new ontology based on the respective template; and finally downloading the new produced ontology.

2) *Discovery of Semantic Relationships.* After executing SDWS tool the resulting ontology has 43 *Operation* instances. Therefore, the total number of comparison pairs between n operations is given by $nc = (43^2 - 43)/2 = 903$. For each comparison pair four semantic similarities are calculated: Wu-Palmer [10], Lin [11], Path [12] and Lesk [13]. For each set of results, the arithmetic mean and standard deviation are calculated in order to obtain the upper and lower thresholds using formulas (3) and (4). A voting schema is used, which considers three possible results: *isSimilarTo*, when the semantic measure value results over the upper threshold; *isDifferentTo*, when the semantic measure value results under the lower threshold; and *Undefined*, when the semantic measure value results between the upper and lower thresholds. The final decision uses a user-defined majority value. If the number of *isDifferentTo* results is greater than the majority value, then the final meaning is established as *isDifferentTo*. If the number of *isSimilarTo* results is greater than the majority value, then the final meaning is established as *isSimilarTo*. For this experimentation the majority value was established in 2.

3) *Instantiate new Semantic Relationships.* The last phase is to assert the new semantic relationships between individuals into the ontology. For this step we are considering only the *isSimilarTo* and *isDifferentTo* results. Figure 2 shows that the operation *get_COMPANY_PROFESSION* *isDifferentTo* 6 other operation instances.

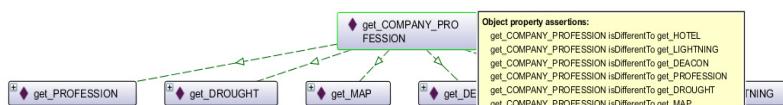


Fig. 2. Some semantic assertions between operations of the *isDifferentTo* relation

6 Evaluation of Results

For evaluation *Precision* and *Recall* measures were calculated [14]. A human (Web service requestor) compared operation pairs and decided based on observation and semantic sound if operation pairs were "*Similar*" or "*Different*". Table 1 shows the *Precision* and *Recall* measures results. From the 903 operation names comparisons, 12 *isSimilarTo* semantic relationships were asserted into the ontology. From these results only 6 were the result of an unanimous vote, the rest of *isSimilarTo* relations are due to the votes of Lin and Lesk measures. From the same set of 903 operation names comparisons, 75 *isDifferentTo* semantic relations were asserted into the ontology. The rest of comparison pairs resulted *unDefined*.

Table 1. Precision and recall results

	Precision	Recall	F-measure
<i>isSimilarTo</i>	1.000000	0.218182	0.358209
<i>isDifferentTo</i>	0.986667	0.087264	0.160347
<i>unDefined</i>	0.945666	0.862028	0.901912

It is important to note that 4 inconsistencies were found between Lin and Path measures, where the upper threshold of Lin establishes them as *isSimilarTo*, whereas Path lower threshold defines them as *isDifferentTo*. These particular cases occur between the operation names *get_HOTEL* and *get_MAP*. With this particular example it is possible to see that Lin is erroneously giving false positives. Wu-Palmer similarity obtains 100 more *isSimilarTo* results than the rest of measures. This indicates that Wu-Palmer measure may be giving more false positives. One of the important features is the secure establishment of similarity relations in the ontology. In the case of finding inconsistencies the voting scheme results in *unDefined* and therefore these relationships are not asserted into the ontology.

7 Conclusions

We have described a semantic enrichment approach which automatically represents any set of available Web service descriptions as ontologies. We have used SDWS, a tool that facilitates the automatic translation of different service description files into ontological models. SDWS incorporates a set of Web service parsers that allow the automatic extraction of service interface definitions for their semantic representation, the main benefit if this tool is that does not require human intervention, facilitating end users to make use of semantic Web technologies without added complexity.

We have described a semantic relatedness discovery process which calculates four semantic similarities between all operations pairs, then calculates the upper and lower thresholds; and identifies operation pairs that are over and under respective thresholds. To assert new semantic relationships between operations into the ontology, a voting scheme is used assuring that the establishment of semantic relations is sufficiently reliable since it is based on a majority vote. In the case of inconsistent results, these are not asserted into the ontology.

Experimentation shows that although few similarity relations are found and asserted, they represent an important step towards the automatic discovery of information that was previously unknown and that can be very useful during automatic search, selection and invocation of Web services based on the operation names. As future work, more similarity measurements will be applied using different approaches, such as: syntactic, semantic, structural and pragmatic. These similarity measurements will be extended to more elements of the Web service descriptions, such as input and output parameter names and parameter types, and texts of the document tags.

References

1. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. *IEEE Intelligent Systems* 16(2), 46–53 (2001)
2. Gruber, T.: A Translation approach to portable ontologies. *Knowledge Acquisition* 5(2), 199–220 (1993)
3. Bravo, M., Pascual, J., Rodríguez, J.: Semantic Representation of Public Web Service Descriptions. In: Murgante, B., Misra, S., Carlini, M., Torre, C.M., Nguyen, H.-Q., Taniar, D., Apduhan, B.O., Gervasi, O. (eds.) *ICCSA 2013, Part V. LNCS*, vol. 7975, pp. 636–651. Springer, Heidelberg (2013)
4. Sycara, K., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of semantic Web services. *Web Semantics: Science, Services and Agents on the World Wide Web* 1(1), 27–46 (2003)
5. Pathak, J., Koul, N., Caragea, D., Honavar, V.G.: A framework for semantic web services discovery. In: Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management, pp. 45–50. ACM (2005)
6. Klusch, M., Fries, B., Sycara, K.: Automated semantic Web service discovery with OWLS-MX. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 915–922. ACM (2006)
7. Gomadam, K., Ranabahu, A., Wu, Z., Sheth, A.P., Miller, J.: A declarative approach using SAWSDL and semantic templates towards process mediation. In: *Semantic Web Services Challenge*, pp. 101–118. Springer, US (2009)
8. Du, X., Song, W., Munro, M.: A Method for Transforming Existing Web Service Descriptions into an Enhanced Semantic Web Service Framework. In: *Information Systems Development*, pp. 217–226. Springer, US (2010)
9. Elgazzar, K., Hassan, A.E., Martin, P.: Clustering wsdl documents to bootstrap the discovery of Web services. In: *2010 IEEE International Conference on Web Services (ICWS)*, pp. 147–154. IEEE (2010)
10. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, pp. 133–138. Association for Computational Linguistics (1994)
11. Lin, D.: An information-theoretic definition of similarity. In: *ICML*, vol. 98, pp. 296–304 (1998)
12. Leacock, C., Chodorow, M.: Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database* 49(2), 265–283 (1998)
13. Banerjee, S., Pedersen, T.: An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. In: Gelbukh, A. (ed.) *CICLing 2002. LNCS*, vol. 2276, pp. 136–145. Springer, Heidelberg (2002)
14. Baeza-Yates, R., Ribeiro-Neto, B.: *Modern information retrieval*, vol. 463. ACM Press, New York (1999)
15. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Sycara, K.: OWL-S: Semantic markup for web services. *W3C member submission*, 22 (April 2004)
16. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Fensel, D.: Web service modeling ontology. *Applied Ontology* 1(1), 77–106 (2005)
17. Kopecky, J., Vitvar, T., Bournez, C., Farrell, J.: SAWSDL: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing* 11(6), 60–67 (2007)

Parameterized Algorithms for Matching and Ranking Web Services

(Short Paper)

Fatma Ezzahra Gmati¹, Nadia Yacoubi-Ayadi¹, and Salem Chakhar^{2,3}

¹ National School of Computer Sciences, University of Manouba, Tunis, Tunisia
`{fatma.ezzahra.gmati,nadia.yacoubi.ayadi}@gmail.com`

² Portsmouth Business School, University of Portsmouth, Portsmouth, UK

³ Centre for Operational Research and Logistics, University of Portsmouth,
Portsmouth, UK
`salem.chakhar@port.ac.uk`

Abstract. The paper presents two parameterized and customizable algorithms for matching and ranking Web services. Given a user query and a set of available Web services, the matching algorithm performs a logic-based semantic matchmaking to select services that functionally match the query and maintains those which fully verify the constraints specified by the user. The ranking algorithm takes as input the matching Web services, assigns to each one a score in the range 0-1 and finally rank them based on the score values. The algorithms have been implemented, evaluated and compared to iSEM and SPARQLent. Results show that the algorithms behave globally well in comparison to these frameworks.

Keywords: Web Service, Service Composition, Semantic Similarity, Matchmaking, Service Ranking.

1 Introduction

Web service matchmaking is a crucial issue within Web service composition [2][3]. It is related to the selection of the most appropriate one among the different candidate Web services. In this paper, we propose two algorithms for matching and ranking Web services. The matching algorithm performs a logic-based semantic matchmaking to select services which fully verify the functional requirements and the constraints specified by the user. The ranking algorithm first assigns to each matching Web service a score in the range 0-1. The scores, along with some user's preference information, are then used to rank the selected Web services.

We developed and implemented a prototype supporting both algorithms. We used the Semantic Matchmaker Evaluation Environment to evaluate the performance of the algorithms and compared them to iSEM [5] and SPARQLent [7]. Results show that our algorithms behave globally well in comparison to iSEM and SPARQLent.

The paper is structured as follows. Sections 2 and 3 detail the matching and ranking algorithms. Section 4 presents the performance analysis. Section 5 discusses some related work. Section 6 concludes the paper.

2 Matching Web Services

2.1 Similarity Measure

The similarity measure quantifies the semantic distance between the two entities participating in the match. The similarity measure, μ , of two service attributes is a mapping that measures the semantic distance between the conceptual annotations associated with the service attributes. It is defined as follows [4]:

$$\mu : A \times A \rightarrow \{\text{Exact, Plug-in, Subsumption, Container, Part-of, Fail}\},$$

where A is the set of all possible attributes. The definitions of semantic measures are given in [2][3][4]. A preferential total order is established on the above mentioned similarity maps: Exact \succ Plug-in \succ Subsumption \succ Container \succ Part-of \succ Fail, where $x \succ y$ means that x is preferred over y . In this paper, we implemented the idea of [1] to compute the similarity degrees.

2.2 Functional Conjunctive Matching

A service S is defined as a collection of attributes that describe the service. Let $S.A$ denotes the set of attributes of service S and $S.A_i$ denotes each member of this set. Let $S.N$ denotes the cardinality of this set. Let S^R be the service that is requested, and S^A be the service that is advertised. A first customization of functional conjunctive matching is to allow the user to specify a desired similarity measure for each attribute. A second customization is to allow the user specifying which attributes should be utilized during the matching process, and the order in which they are considered. These two customizations can be supported through the concept of Criteria Table [4]. A Criteria Table, C , is a relation consisting of two attributes, $C.A$ and $C.M$. $C.A$ describes the service attribute to be compared, and $C.M$ gives the *least* preferred similarity measure for that attribute. Let $C.A_i$ and $C.M_i$ denote the service attribute value and the desired measure in the i th tuple of the relation. Let $C.N$ be the number of tuples in C .

Based on the concept of Criteria Table, a sufficient functional conjunctive match between services is defined as follows:

$$\begin{aligned} & \forall i \exists_{j,k} (C.A_i = S^R.A_j = S^A.A_k) \wedge \mu(S^R.A_j, S^A.A_k) \succeq C.M_i \\ & \Rightarrow \text{SuffFuncConjMatch}(S^R, S^A) \quad 1 \leq i \leq C.N. \end{aligned} \tag{1}$$

This basic matching rule is extended in [2][3] to support functional disjunctive, functional generic, service-level and quality of service-based matching.

2.3 Matching Algorithm

The matching algorithm that follows from Sentence (1) is given in Algorithm 1. It loops on the available Web services set U and for each service S^A it proceeds as follows: (i) scans the Criteria Table and for each attribute identifies the corresponding attributes in S^R and S^A ; (ii) computes the similarity degrees between

the corresponding attributes; and (iii) appends the service S^A and the corresponding similarity degree in the list `mServices` of matching Web services. The output of Algorithm 1 is the list `mServices`. Algorithm 1 applies to functional conjunctive matching. It can be extended to apply to other types of matching.

Algorithm 1. Matching

```

Input :  $S^R$ : Requested service;  $C$ : Criteria Table.
Output: mServices: Matching services.
1  $U \leftarrow \{\text{list of potential advertised services}\};$ 
2 for (each service  $S^A$  in  $U$ ) do
3   while ( $i \leq C.N$ ) do
4     while ( $j \leq S^R.N$ ) do
5       if ( $S^R.A_j = C.A_i$ ) then
6         Append  $S^R.A_j$  to rAttrSet;
7          $j \leftarrow j + 1;$ 
8       while ( $k \leq S^A.N$ ) do
9         if ( $S^A.A_k = C.A_i$ ) then
10          Append  $S^A.A_k$  to aAttrSet;
11          $k \leftarrow k + 1;$ 
12        $i \leftarrow i + 1;$ 
13      $bol \leftarrow \text{true};$ 
14     while ( $t \leq C.N \wedge bol$ ) do
15        $mu \leftarrow \mu(\text{rAttrSet}[t], \text{aAttrSet}[t]);$ 
16       if ( $mu \geq C.M_1$ ) then
17         simDegrees[ $t$ ]  $\leftarrow mu;$ 
18       else
19          $bol \leftarrow \text{false};$ 
20        $t \leftarrow t + 1;$ 
21     if ( $bol$ ) then
22       Append ( $S^A, \text{simDegrees}[1], \dots, \text{simDegrees}[C.N]$ ) to mServices;
23 return mServices;

```

Inferring $\mu(\cdot, \cdot)$ by ontological parse of pieces of information into facts and then utilizing fast rule-based engines leads to $O(|R||F||P|)$ where $|R|$ is the number of rules, $|F|$ is the number of facts, and $|P|$ is the average number of patterns in each rule [4]. Hence, the complexity of Algorithm 1 is $O(|U|C.N \times S^A.N) + O(|U||R||F||P|) \simeq O(|U||R||F||P|)$ (since, as underlined by [4], the process of computing μ is the most “expensive” step of the algorithm).

3 Ranking Web Services

3.1 Computing of Web Services Scores

The scores of the Web services are computed based on the similarity measures of the matching attributes specified in the Criteria Table. We need to assign a numerical weight to every similarity degree as follows: Fail: w_1 , Part-of: w_2 ,

Container: w_3 , Subsumption: w_4 , Plug-in: w_5 and Exact: w_6 . In this paper, we assume that the weights are computed as follows:

$$w_1 \geq 0, \quad (2)$$

$$w_i = (w_{i-1} \cdot p) + 1, \quad i = 2, \dots, p; \quad (3)$$

where p is the number of attributes. This way of weights computation ensures that a single higher similarity degree will be greater than a set of p similarity degrees of lower weights taken together. Indeed, the weights values verify the following condition: $w_i > pw_j, \forall i > j$. Then, the initial score of an advertised service S^A is computed as follows:

$$\rho(S^A) = \sum_{i=1}^{i=p} w_i. \quad (4)$$

The scores given by Equation (4) are not in the range 0-1. The following equation can be used to normalize the scores (U' is the set of matching Web services):

$$\rho'(S^A) = \frac{\rho(S^A) - \min_{K \in U'} \rho(S^K)}{\max_{K \in U'} \rho(S^K) - \min_{K \in U'} \rho(S^K)}. \quad (5)$$

3.2 Ranking Rule

The basic information used for ranking Web services are the scores. However, the use of the scores only may lead to the problem of ties. To avoid this problem, we may exploit the user's preference information given in the Criteria Table. Two measures can be used to avoid the problem of ties:

- *Difference between the desired similarities.* Let s be a similarity degree and denote by $Ord(s)$ the ordinal rank of s in respect to the other predefined similarity degrees. The definition of $Ord(\cdot)$ is as follows: Fail: 1, Part-of: 2, Container: 3, Subsumption: 4, Plug-in: 5 and Exact: 6. Let S^A be an advertised service and let (s_1, \dots, s_p) be its similarity degrees for attributes A_1, \dots, A_p . Then, we define the function $Diff(\cdot, \cdot)$ as follows:

$$Diff(S^A, C) = \sum_{k=1}^{k=p} \{Ord(s_k) - Ord(C.M_k)\}. \quad (6)$$

- *Lexicographic selection.* Let S_i^A and S_j^A be two services and let (s_1, \dots, s_p) and (s'_1, \dots, s'_p) be their similarity degrees for attributes A_1, \dots, A_p . Then, the lexicographic rule is defined as follows:

$$Lex(S_i^A) > Lex(S_j^A) \Leftrightarrow (\exists l)(\forall r < l)(s_r = s'_r) \wedge (s_l > s'_l). \quad (7)$$

The ranking rule is then stated as follows: use the score-based ranking and if there is a tie, apply the order-based ranking; if another tie occurs, apply the lexicographic selection; and if a further tie occurs, select the first service. Formally,

```

RR: if  $\rho'(S_i^A) > \rho'(S_j^A)$  then  $S_i^A \succ S_j^A$ 
    else if  $\rho'(S_i^A) = \rho'(S_j^A)$  then
        if  $Diff(S_i^A, C) > Diff(S_j^A, C)$  then  $S_i^A \succ S_j^A$ 
        else if  $Diff(S_i^A, C) = Diff(S_j^A, C)$  then
            if  $Lex(S_i^A) > Lex(S_j^A)$  then  $S_i^A \succ S_j^A$ 
            else if  $Lex(S_j^A) = Lex(S_i^A)$  then
                else Select the first service

```

Assume that the matching algorithm has identified the services given in Table 1. This table also shows the initial and normalized scores computed through Equations (4) and (5), respectively. Table 2 summarizes the ranking process of services given in Table 1. The final ranking has been obtained by using successively the scores, the similarity difference and the lexicographic selection.

Table 1. Web services identified by the matching algorithm and their scores

Service S_i	Input	Output	Service category	$\rho(S_i)$	$\rho'(S_i)$
S_1	Exact	Subsume	Subsume	147	0.5
S_2	Plug-in	Exact	Subsume	174	1
S_3	Plug-in	Plug-in	Plug-in	120	0
S_4	Plug-in	Subsume	Exact	174	1
S_5	Subsume	Exact	Subsume	120	0.5

Table 2. Ranking process

Rule	Test	Ranking
Scores	$\rho'(S_i^A) > \rho'(S_j^A)$	$S_2 = S_4 \succ S_1 = S_5 \succ S_3$
Order difference	$\rho'(S_i^A) = \rho'(S_j^A) \wedge Diff(S_i^A, C) > Diff(S_j^A, C)$	$S_2 = S_4 \succ S_1 = S_5 \succ S_3$
Lexico. selection	$Diff(S_i^A, C) = Diff(S_j^A, C) \wedge Lex(S_i^A) > Lex(S_j^A)$	$S_2 \succ S_4 \succ S_1 \succ S_5 \succ S_3$

3.3 Ranking Algorithm

The proposed ranking process is given in Algorithm 2. It takes the list of matching services `mServices` as input and proceeds as follows: (i) computes, for each matching service, the initial score using function `ComputeScore`, which implements Equation (4); (ii) computes the minimum and maximum scores values and then, for each matching service, it computes the normalized score using Equation (5); and (iii) uses the ranking rule `RR` to sort Web services. The output of Algorithm 2 is an ordered list `mServices` of matching Web services. We note that $N + 2$ in Algorithm 2 indicates the index of the score value in `mServices`.

The complexity of the first *while* loop of Algorithm 2 is $O(pC.N)$. The computing of the minimum or maximum values of a list of n values is $O(n)$. The complexity of the two last *while* loops, which correspond to a comparison-based sorting algorithm, is at best $O(n \log n)$ and in worst case, it makes $O(n^2)$. Hence, the overall complexity of Algorithm 2 in best case is $O(pC.N) + O(2n) + O(n \log n)$ and in worst case is $O(pC.N) + O(2n) + O(n^2)$.

Algorithm 2. Ranking

Input : $mServices$: Services list; N : Nb. of attributes; C : Criteria table; w : weights vector.
Output: $mServices$: Ranked list of matching services.

```

1  $r \leftarrow \text{length}(mServices);$ 
2 while ( $t \leq r$ ) do
3    $row \leftarrow t$ th row in  $mServices;$ 
4    $s \leftarrow \text{ComputeScore}(row, w);$ 
5   Append  $s$  to  $row$ ;
6   Append  $row$  to  $mServices$ ;
7    $a \leftarrow \min(mServices);$ 
8    $b \leftarrow \max(mServices);$ 
9   while ( $l \leq r$ ) do
10     $ns \leftarrow (mServices[l, N + 2] - a)/(b - a);$ 
11     $mServices[l, N + 2] \leftarrow ns;$ 
12   while ( $i \leq r$ ) do
13     while ( $j \leq r$ ) do
14        $row_i \leftarrow \text{row } i \text{ in } mServices;$ 
15        $row_j \leftarrow \text{row } j \text{ in } mServices;$ 
16       if ( $\text{RR}(row_i[1], row_j[1], C, w)$ ) then
17          $tmp \leftarrow row_j;$ 
18          $row_j \leftarrow row_i;$ 
19          $row_i \leftarrow tmp;$ 
20 return  $mServices;$ 
```

4 Performance Analysis

We implemented a prototype called PMRF (Parameterized Matching-Ranking Framework) that supports both the matching and ranking algorithms. We used OWLS API to parse the published Web services list and the user request. The similarity between the concepts is inferred using Jena API. In the current version, PMRF supports only Input and Output attributes. The open source tool SME2 (Semantic Matchmaker Evaluation Environment) has been used for performance evaluation and comparison using OWLS-TC collections. The implemented Plugin for the experiment requires a precise interface. Hence, we assigned to the Criteria Table the default values (Input: Fail, Output:Fail).

We compared PMRF with SPARQLent [7] and iSEM [5] frameworks. The results of analysis are shown in Fig. 1. The test collection used is OWLS-TC4, which consists of 1083 service offers described in OWL-S 1.1 and 42 queries. Fig. 1.a shows that PMRF recall is significantly better than iSEM logic-based and SPARQLent. This means that PMRF is able to reduce the amount of false positives. Fig. 1.b indicates that PMRF has a more accurate precision than iSEM logic-based and SPARQLent. This is due to the use of the score-based ranking, which gives a more coarse evaluation than degree-based aggregation. Fig. 1.c attests that PMRF is faster than SPARQLent and slightly less faster than iSEM. SPARQLent has a high query response time if the query include preconditions/effects. PMRF and iSEM have close query response time because both consider only direct parent/child relations. Finally, Fig. 1.d reveals that PMRF consumes less memory than iSEM logic-based and SPARQLent.

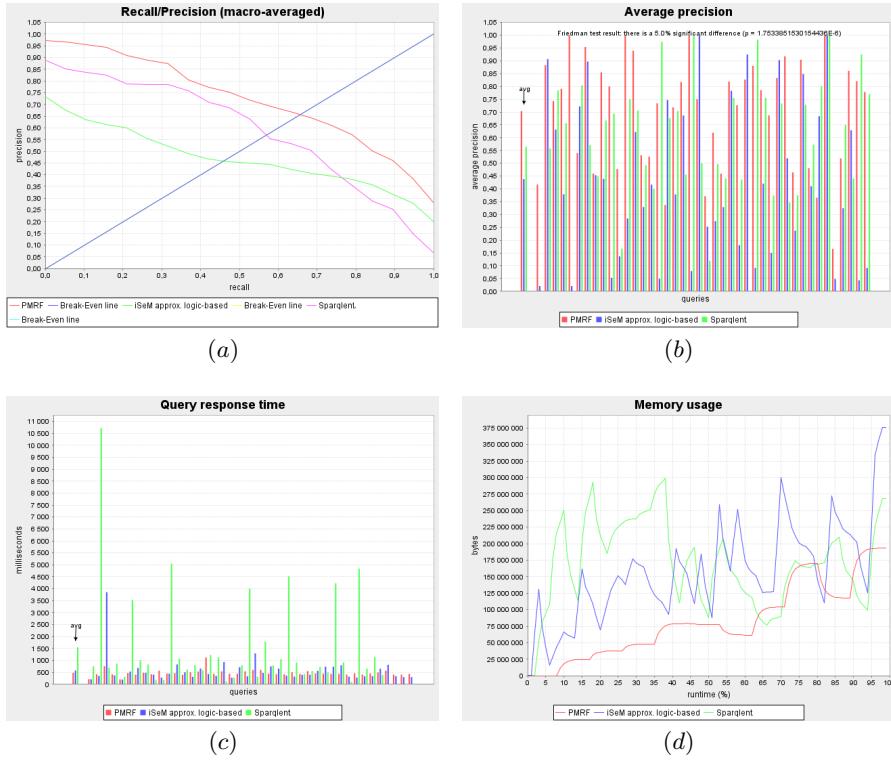


Fig. 1. Results of performance analysis

5 Related Work

A parameterized semantic matchmaking framework that enables the user to specify the matched attributes and the order in which attributes are compared is proposed in [4]. In [1], the authors propose a greedy-based algorithm that relies on the concept of matching bipartite graphs. The frameworks iSEM [5] and SPARQLent [7] consider preconditions and postconditions of Web service. In [2][3], we extended the work of [4] by relaxing matchmaking conditions.

Different ranking algorithms have been proposed. In [6], the authors propose a ranking method that combines the quality of service and fuzzy logic. The authors in [8] provide a context based method where the final rank list is obtained based on the proximity of the context of similar Web services. In [9], the authors use the multicriteria analysis to sort Web services based on the dominance scores.

In this paper, we improved our previous matching algorithms given in [2][3] and added a new algorithm for ranking Web services. Although our approach relies entirely on logical techniques, it integrates a scoring technique, which provides a ranked list of discovered services.

6 Conclusion

We presented two parameterized algorithms for Web service matching and ranking. The matching algorithm takes as input a user query and performs a logic-based semantic matchmaking in order to select the services that match the user's constraints. Then, the ranking algorithm sorts the matching services based on their scores and the user's preference information. Both algorithms have been implemented and the performance analysis shows that the algorithms behave globally well in comparison to SPARQLent [7] and iSEM [5]. In the future, we intend to: (i) finalize the development of PMRF to support all attributes and different matching types, (ii) use other sorting algorithms, and (iii) extend our work to quality of service-based matching.

References

1. Bellur, U., Kulkarni, R.: Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching. In: IEEE International Conference on Web Services, pp. 86–93. IEEE Computer Society, Los Alamitos (2007)
2. Chakhar, S.: Parameterized Attribute and Service Levels Semantic Matchmaking Framework for Service Composition. In: Fifth International Conference on Advances in Databases, Knowledge, and Data Applications, pp. 159–165. IARIA - International Academy, Research, and Industry Association, Wilmington (2013)
3. Chakhar, S., Ishizaka, A., Labib, A.: QoS-Aware Parameterized Semantic Matchmaking for Web Service Composition. In: The 10th International Conference on Web Information Systems and Technologies, pp. 50–61. SciTePress - Science and Technology Publications, Barcelona (2014)
4. Doshi, P., Goodwin, R., Akkiraju, R., Roeder, S.: Parameterized Semantic Matchmaking for Workflow Composition. IBM Research Report RC23133, IBM Research Division (2004)
5. Klusch, M., Kapahnke, P.: The iSEM matchmaker: A Flexible Approach for Adaptive Hybrid Semantic Service Selection. Journal of Web Semantics 15, 1–14 (2012)
6. Manoharan, R., Archana, A., Cowla, S.N.: Hybrid Web Services Ranking Algorithm. International Journal of Computer Science Issues 8, 83–97 (2011)
7. Sbodio, M.L., Martin, D., Moulin, C.: Discovering Semantic Web Services Using SPARQL and Intelligent Agents. Journal of Web Semantics 8, 310–328 (2010)
8. Segev, A., Toch, E.: Context Based Matching and Ranking of Web Services for Composition. IEEE Transactions on Services Computing 3, 210–222 (2011)
9. Skoutas, D., Sacharidis, D., Simitsis, A., Sellis, T.: Ranking and Clustering Web Services Using Multicriteria Dominance Relationships. IEEE Transactions on Services Computing 3, 163–177 (2010)

Arabic Domain Terminology Extraction: A Literature Review (Short Paper)

Ibrahim Bounhas^{1,2}, Wiem Lahbib¹, and Bilel Elayeb^{3,4}

¹LISI Laboratory of computer science for industrial systems, Carthage University, Tunisia
Bounhas.Ibrahim@gmail.com

²Higher Institute of Documentation (ISD), Manouba University, 2010 Tunisia
wiemlahbib88@hotmail.fr

³RIADI Laboratory, The National School of Computer Science (ENSI),
Manouba University 2010 Tunisia.

⁴Emirates College of Technology, P.O. Box: 41009. Abu Dhabi, United Arab Emirates
Bilel.Elayeb@riadi.rnu.tn

Abstract. Domain terminology extraction is an important step in many applications such as ontology building and information retrieval. Analyzing a corpus to automatically extract key terms is a difficult task, especially in the case of Arabic language. The complexity of spelling, morphology and semantics of Arabic makes natural language processing tasks quite difficult. In addition to the complexity of Arabic, the challenges related to domain terminology extraction are caused by the inherent difficulty in determining whether a word or a phrase represents or not a given text. All these problems have not restricted the multitude of Arabic terminology extraction approaches in the ontology building process. Therefore, this article presents a literature review in the field of Arabic terminology extraction focusing on the specificities of this language.

Keywords: Terminology extraction, Natural Language Processing, Arabic language, Multi-Word Terms.

1 Introduction

The terminology extraction represents a critical step in the ontologies building process which requires the performance of several complex tasks and various pretreatments. Indeed, the conceptualization demands the recognition of key terms representing concepts. The passage of the word-concept is a complex process that includes the extraction of pertinent terms and semantic relationships [1].

The extraction of concepts requires several treatments that affect all levels NLP. Indeed, the concepts are represented by linguistic units which can be single words or phrases [2].

However, all these processes become more challenging when it concerns the Arabic language, since it has many characters that can be considered as sources of ambiguity

having a direct influence on NLP in general and particularly on the extraction of terminologies. Firstly, this language is characterized by non-alphabetic signs called diacritics or short vowels that can be added above or below the letters. The phenomenon of “vocalization” (تشكيل) is sometimes necessary to interpret a word. However, the arabic vowelized texts are rare. And so, the absence of short vowels generates several cases of lexical and morphological ambiguity.

Second, Arabic is characterized by several types of variations. In spelling, some similar traits and characters are confusing in scripting. This is the case of the letters “ا”, “أ” and “إ” and couples (ء, ئ) and (ي, ي). Morphologically, Arabic is a derivational language and words are generated from roots using patterns. At the syntactic level, we can change the order of words of a sentence or an expression in a quasi-random manner. For example, the phrase “رجل في البيت” (a man in the house) is equivalent to the phrase “في البيت رجل” (in the house, there are a man).

The objective of this paper is to study the problem of Arab Terminologies extraction (ATE) and provide a comparative review of existing approaches. Indeed, it is a research field that is largely unexplored, at least if we compare ourselves to other languages such as Latin origin’s [3]. To facilitate the study of these approaches, we begin by defining the terminology and introducing some terminologies extraction tools (see section 2) and the key elements of the ATE approaches (see section 3). Existing approaches will be discussed based on these elements (see section 4). However, we do not claim completeness, since some terminology extraction works are published as sub-elements of larger works such as IR and ontologies applications.

2 Terminology and ATE Tools

In this review, we focus on NLP aspects of Arabic language that affects the terminology extraction process. However, recognition of linguistic units is not enough to extract terminology and other treatments are needed. To understand these treatments, it is necessary to define certain concepts.

We can conclude that a domain is a knowledge sector composed of related items (products). According to [4], *terminology* is a set of names and notions. The terminology can be defined as the set of terms representing a specific area. Jacquemin [5] defines a *term* as a surface representation of a concept bellowing to one specific domain.

In this review, we identify ATE approaches and tools. These approaches can be classified into three categories according to the type of knowledge they use (language and/or statistics). Whether we face linguistic or hybrid approach, NLP tools are integrated. In what follows in this section, we are going to identify the main tools used by existing work. We classify these tools as follows.

- Segmentation tools (tokenization): segmenting the text fragments into sentences and finally into words. The ArabicSVMTools package [6] for example, contains a tool that splits a sentence into words.
- The Part Of Speech tagging tools (POS tagger): identify the part of speech of each word using context (words in left and words in right). These tools are built by learning in labeled corpus. We quote the following tools: Diab POS tagger [7],

- AlGahtani POS tagger [8], Khoja POS tagger¹ and Stanford POS Tagger². In what follows we use POS to mean “Part Of Speech” and POS Tagger for “Part Of Speech labeling tool”
- Stemming tools: identify the lemma or root of a word. Among those used in cited works in this review, we enumerate: Khoja stemmer³ and El-Beltagy Stemmer [9].
 - Morphological analyzers: analyze a word structure and recognize its morphological characteristics. Some studies indicated in this paper use AraMorph [10].
 - Morphological disambiguation tools: disambiguate a word morphologically choosing the right morphological solution from a list provided by a morphological analyzer. These tools return, in addition to POS, morphological attributes of a treated word such as gender and number for nouns and verbs mode. We name the MADA tool [11] and the tool of Ayed [12].
 - Parsers: analyze the structure of a sentence or phrase and recognize the roles of its constituents. Some studies cited in this review use Stanford Parser⁴. Other tools that allow the implementation of patterns or rules combined with or without measures of association are used to extract compound terms. We name GATE⁵ and KP-Miner [13].

3 ATE Approaches

This section provides an overview of the main existing approaches of arabic terminology extraction. These approaches are classified in three categories: domain-specific approaches, semi-specific approaches and general approaches.

3.1 Domain-Specific Approaches

Some works relating to the ontologies building process, are based on the derivation to select candidate term or to add relationships. In this vein, Zaidi and al. [14] have proposed to build a legislative ontology in purpose to be exploited in IR and particularly in the query reformulation. The process of ontology building is based on the top-down strategy for the hierarchy of concepts in the first place, starting with the two principal concepts “legal system” and “civil law”. Then, for every concept, they combine a set of closest synonyms to the legislative domain.

In terms of processing multilingual corpus (Arabic-English), Yousif and al. [15] proposed a term extraction approach. They built a system called LOLO which is based on a comparison of the relative frequencies calculated from specific corpus against general corpus frequencies. To achieve this goal, they began by treating the

¹ <http://zeus.cs.pacificu.edu/shereen/research.htm#tagging>

² The Stanford Arabic POS Tagger,

<http://nlp.stanford.edu/software/tagger.shtml>

³ <http://zeus.cs.pacificu.edu/shereen/research.htm#stemming>

⁴ <http://nlp.stanford.edu/software/lex-parser.shtml>

⁵ <http://gate.ac.uk/sale/lrec2000/lrecmain.html>

text through a segmentation performed using the “Arabic word segmenter” tool part of the package “ArabicSVMtools”.

In the similar vein, Boulaknadel and al. [16] presented a hybrid approach to extract compound words from a corpus representing the domain of the environment. They identified patterns that exploit the recognized POS using Diab POS Tagger. Statistical measurements are used, thereafter, to filter the candidate terms. In their experiments, they used a collection of 1,062 documents containing 475,148 words. The best result was recorded using LLR, with an accuracy of 85%.

Recently, El Mahdaouy and al. [17] presented a similar approach to [16], using the same labeling patterns and tool. However, the authors have given particular attention to the treatment of variations to improve the recall. Thus, they treated spelling variants, inflectional variants, variants morpho-syntactic and syntactic variants of candidate terms. Subsequently, the first N compound terms are compared against a reference list extracted from AGROVOC. As a result, NLC gave the best precision (82% against 75% returned by LLR).

In another attempt to deal with Arabic language ambiguities, Bounhas and al. [2] have tried to solve the morphological and syntactic ambiguities and assesses these two dimensions in one single phase. To do so, the authors have kept the morphological attributes returned by MADA, having a standardized score higher than 0,7. This approach can be summarized as follow. First, the authors carry out a full morphological and syntactic analysis of expressions. Second, the authors used several corpus representing different areas to assess the termhood propriety. Thirdly, they employed a possibilistic network to evaluate the two dimensions. Proceeding from “MADA+IF-IDF+LLR” approach to the possibilistic approach, the authors reported an improvement of 26,67% in of F-measure to reach 63,10%.

In order to build ontologies from Arabic texts, Mazari and al. [18] preferred to use a hybrid approach combining linguistic and statistical filters. The first step consists to prepare the corpus. Indeed, it's about a normalization based on the elimination of numbers, symbols, Roman words, stop words and abbreviations on one hand, and the spelling normalization on the other hand. Thereafter, the authors conducted a light stemming removing some prefixes and suffixes.

3.2 General Approaches

In the same context of [14], Belkredim and al. [19] proposed an approach based on the rules of derivation and inflection. The authors estimated that verbs are often used as root to derive nouns and other verbs.

The ontology is then, formed by nouns derived from six categories (verbal noun, active participle, passive participle, background adjectives, nouns of time and space and instrument nouns) and derived from the trilateral root verbs and quadrilaterals. Regardless of selecting derivatives manually [14] or automatically [19], this approach does not define the termhood, even if the approach is applied to a specific domain.

The KP-Miner system has been used by many researchers; among them we strike on [13] who proposed to extract simple and compound words from English and Arabic general and specific documents. First, stop words and punctuation are removed.

In a first experiment, the authors used 100 Wikipedia articles and compared their tool against the Sakhr tool⁶. It achieved 32,3% as a recall rate and 6% as a reported improvement over this tool. In a second experiment, El-Beltagy and Rafea [13] used 18 agricultural documents containing 53,696 words.

Further to the use of Wikipedia, Attia and al. [20] have also integrated Arabic WordNet for extracting compound terms in addition to two other different methods. The first method aims at extracting non-decomposable terms using certain indicators. Compounds tiles of Arabic Wikipedia served as candidate terms translated later into 21 languages⁷. If the title is in MINLEX⁸ or translated by a single word or found in WordNet, it is considered then as a compound term. The second method aims to collect words from english WordNet and translate them using Google translate. The third method uses the POS-based patterns provided by the MADA tool and measures of association for assessing unithood.

El- Shishtawy and Al-Sammak [21] had uses machine learning technique to extract simple and compound candidate terms. The approach includes the elimination of stop words, the calculation of frequencies, stemming using Khoja stemmer and POS tagging using Khoja POS tagger. To extract compound terms, the authors are limited to expressions composed of three words and applied POS-based rules and lemmas. In the automatic evaluation, the authors are compared against KP-Miner and Sakhr tools. The best results were obtained as follows: i) In the second collection, P-10=0,53 and R-5=0,56; and, ii) in the third series, P-10=0,65; R-7=0,46.

3.3 Semi-Specific Approaches

Interested by religious texts, Mashaan Abed and al. [22] presented an approach for extraction of simple and compound terms. They began by a pretreatment of their corpus extracted from Islamic websites (including shamelा.ws and islamweb.net). These pretreatments include the elimination of punctuation, spelling standardization, segmentation of the text into words and disposal of stop words. The extraction is carried out simply using TF-IDF. The extraction of compound terms is more complex and includes several steps. Assuming that a compound term is composed of a head and an expansion, the authors started from the extraction of the heads, which are the words having a number of occurrences greater than 1.

In terms of the combining both linguistic and statistical techniques, Zaidi and al. [23] presented a hybrid approach to extract simple and compound words from the Quran. Regarding simple terms, they began by morphological analysis using Ara-Morph. Thereafter, they weighted terms by using two techniques: i) comparing the frequencies of terms against a general corpus; and ii) applying TF-IDF on documents of a specialized corpus.

⁶ <http://www.sakhr.com/Technology/Keyword/Default.aspx?sec=Technology&item=Keywords>

⁷ Among these languages: German, Greek, English, Spanish, French, Indonesian, Italian, Polish, Romanian, Russian, Swedish and Turkish.

⁸ Multi-language glossary

4 Synthesis

In this section we try to summarize the ATE approaches, studying their main characteristics.

From the type of corpus perspective, the majority of approaches have used modern corpuses. The exception is made by works applied on religious texts such as the Quran [23], the hadith [2] and Islamic sites [22]. The test corpora of [23] contain documents porting on the Arabic language, some of which are modern, but we cannot judge the whole corpus, since the authors did not cite any sources. We also note that the majority of the corpuses are not vocalized. Some work as [2] used vocalized corpus, but short vowels are not taken into account in the morphological analysis. This means that several ambiguities are induced, which decreases the accuracy of the extracted terms. From the domain perspective, several works have been applied on the environmental and agricultural areas.

Finally, we did not find any approach that use Arabic dictionaries for extracting terminology. Noting that dictionaries may be important sources of knowledge, yet, they are not exploited regarding the Arabic language. However, an effort of structuring and standardization is necessary to make them usable for automatic processing. We also noticed that the document structure have been used only in few works such as [2], although some corpuses such as Wikipedia contain semi-structured documents. In fact, the hierarchical structure of documents can be a rich source of useful contextual information for disambiguation.

From the linguistic analysis perspective, we note several limitations mainly related to the use of a superficial analysis or the lack of a disambiguation phase. We even find researches that don't conduct any analysis as [15] who did not even lemmatized words of their corpus. This also affects the system's ability to extract simple and compound terms.

We note that some authors ([2]) have tried to treat syntactic ambiguities. However, in other works, these ambiguities are addressed indirectly through measures of association, and then it is possible to generate syntax trees. For example the expression “الشرب من قح الماء” (Drinking a glass of water) can be analyzed in two ways: (N+PREP+N) +N or N+ PREP+ (N+N) with N: Name and PREP: preposition. The first alternative is to combine the first three words as a prepositional phrase (الشرب من قح, drinking a glass). The second alternative is to first consolidate the last two words as an annexation phrase (قح الماء, glass of water) before processing the preposition.

From an evaluation perspective, cited works differ widely. Some approaches are not valued as those of [16] and [23]. The majority of works is evaluated only in terms of accuracy, since the automatic calculation of recall requires the existence of a reference list. That is why these works are based on a manual validation limited sometimes to the first N terms. By cons, works using other measures such as Recall and F-measure are based on existing reference lists (such as the thesaurus AGROVOC) or manually constructed (as in [2]). A partial solution is to integrate the terminology in an application and test its effect, as proposed [2].

5 Conclusion

In this article we have tried to study a leading step in the ontology building, named the ATE process through a representative research works. It is clear that the characteristics of the language and the strong dependence of the different NLP levels influence this process. With the lack of standardized resources and poor collaboration between researchers, it is difficult to assess objectively these works. For recommendations, we believe that future projects should incorporate terminology extraction tools that allow a full morphological and disambiguation analysis such as MADA and the Ayed tool [12]. At the syntactic level, we recommend implementing a grammar of noun phrases, taking inspiration from works of Attia and Bounhas. We must also think to standardize (or at least to publish online) resources such as the list of stop words and lists of references to be able to compare and to be compared.

References

1. Bounhas, I., Elayeb, B., Evrard, F., Slimani, Y.: ArabOnto: Experimenting a new distributional approach for building arabic ontological resources. *International Journal of Metadata, Semantics and Ontologies (IJMSO)* 6(2), 91–95 (2011)
2. Bounhas, I., Elayeb, B., Evrard, F., Slimani, Y.: Organizing contextual knowledge for arabic text disambiguation and terminology extraction. *Knowledge Organization* 38(6), 473–490 (2011)
3. Bougouin, A.: État de l'art des méthodes d'extraction automatique de termes-clés. *Actes de la conférence TALN-RECITAL*, 17-21 Juin, Sables d'Olonne, France (2013)
4. Rey A.: La terminologie: noms et notions. France : Presses. Université de France (No. 1780) (1979)
5. Jacquemin, C.: Variation terminologique: Reconnaissance et acquisition automatiques de termes et de leurs variantes en corpus. *Thèse d'habilitation*, Université de Nantes, France (1997)
6. Roberts, A., Al-Sulaiti, L., Atwell, E.: aConCorde: Towards an open-source, extendable concordancer for Arabic. *Corpora* 1(1), 39–60 (2006)
7. Diab, M.T., Kadri, H., Jurafsky, D.: Automatic tagging of arabic text: From raw text to base phrase chunks. In: *Proceedings of The 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, Boston, USA, May 2-7, pp. 149–152 (2004)
8. AlGahtani, S., Black, W., Mc-Naught, J.: Arabic part-of-speech-tagging using transformation-based learning. In: *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools*, April 22-23, pp. 66–70. The MEDAR Consortium, Cairo (2009)
9. El-beltagy, S.R.: A Framework for the Rapid Development of Dictionary Based Domain Specific Arabic Stemmers, *Rapport Technique (TR/COE_WM/12/12/2007)*, Center of Excellence for Data Mining and Computer modeling, Egypt (2007)
10. Hajic, J., Smrz, O., Buckwalter, T., Jin, H.: Feature-based tagger of approximations of functional arabic morphology. In: *The Fourth Workshop on Treebanks and Linguistic Theories*, December 9-10, pp. 53–64. University of Barcelona, Spain (2005)

11. Roth, R., Rambow, O., Habash, N., Diab, M., Rudin, C.: Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. In: Proceedings of the Association for Computational Linguistics conference (ACL), Columbus, Ohio, USA, pp. 117–120 (2008)
12. Ayed, R., Bounhas, I., Elayeb, B., Evrard, F.: Benllamine Ben Saoud N. Arabic Morphological Analysis and Disambiguation Using a Possibilistic Classifier. In: Intelligent Computing Theories and Applications, Proceedings of the 8th International Conference on Intelligent Computing (ICIC), China, pp. 274–279 (2012)
13. El-beltagy, S.R., Rafea, A.: KP-Miner: A keyphrase extraction system for English and Arabic documents. *Information Systems* 34(1), 132–144 (2001)
14. Zaidi, S., Laskri, M.T., Bechkoun, K.: A Cross-language Information Retrieval Based on an Arabic Ontology in the Legal Domain. In: Signal Image Technology and Internet based-systems, Yaoundé Cameroun (2005)
15. Yousif, A., Khurshid, A.: LoLo: A System for Extracting Statistical Information and Lexical Resources from Arabic Corpora. In: Proceedings of the Workshop on Arabic Natural Language Processing, Information and Communication Technologies International Symposium, Fes, Morocco, July 2007, pp. 394–398. IEEE (2007)
16. Boulaknadel, S., Daille, B., Aboutajdine, D.: A multi-word term extraction program for arabic language. In: Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC), Marrakech, Morocco, May 17–23, pp. 1485–1488 (2008)
17. El Mahdaouy, A., El Alaoui Ouatik, S., Gaussier, E.: A Study of Association Measures and their Combination for Arabic MWT Extraction. In: 10th International Conference on Terminology and Artificial Intelligence, Paris, France (2013)
18. Mazari, A.C., Aliane, H., Alimazighi, Z.: Automatic Construction of Ontology from Arabic Texts. In: Proceedings of International Conference on Web and Information Technologies ICWIT, pp. 193–202 (2012)
19. Belkredim, F.Z., El Sebai, A.: An Ontology Based Formalism for the Arabic Language Using Verbs and their Derivatives. *Communications of the IBIMA* 11(5), 44–52 (2009)
20. Attia, M., Toral, A., Tounis, L., Pecina, P., Van Genabith, J.: tomatic Extraction of Arabic Multiword Expressions. In: Proceedings of the 2010 Workshop on Multiword Expressions: from Theory to Applications, Beijing, China, pp. 19–27 (2010)
21. El-shishtawy, T., Al-sammak, A.: Arabic Keyphrase Extraction using Linguistic knowledge and Machine Learning Techniques. In: Proceedings of the Second International Conference on Arabic Language Resources and Tools, the MEDAR Consortium, Cairo, Egypt (2012)
22. Mashaan Abed, A., Tiun, S., Albared, M.: Arabic term extraction using combined approach on Islamic document. *Journal of Theoretical and Applied Information Technology* 58(3), 601–608 (2013)
23. Zaidi, S., Abdelali, A., Laskri, M.T., Al Shenify, M.: Extracting Simple and Compound Terms from Arabic Texts: An Application on The Quranic Text. *Communications of the Arab Computer Society* 4(1) (2011)

Erratum: Flexible Querying for SPARQL

Andrea Calì^{1,2}, Riccardo Frosini¹, Alexandra Poulovassilis¹,
and Peter T. Wood¹

¹ Dept. of Computer Science and Inf. Systems, Birkbeck University of London, UK

² Oxford-Man Institute of Quantitative Finance, University of Oxford, UK

{andrea, riccardo, ap, ptw}@dcs.bbk.ac.uk

R. Meersman et al. (Eds.): OTM 2014, LNCS 8841, pp. 473–490, 2014.

© Springer-Verlag Berlin Heidelberg 2014

DOI 10.1007/978-3-662-45563-0_52

The acknowledgment of the paper starting on page 490 of this volume is missing. It should read:

Acknowledgments. Andrea Calì acknowledges support by the EPSRC project “Logic-Based Integration and Querying of Unindexed Data” (EP/E010865/1).

The original online version for this chapter can be found at
http://dx.doi.org/10.1007/978-3-662-45563-0_28

Author Index

- Agrawal, Sweety 623
Ali, Muhammad Intizar 491
Almendros-Jiménez, Jesús M. 457, 509
Aloia, Nicola 768
Amann, Bernd 130
Athanasiou, Spiros 553
- Baesens, Bart 345
Bahria El Asghar, Nihed 313
Bellatreche, Ladje 760
Bianchini, Devis 364, 517
Bounhas, Ibrahim 745, 792
Bravo, Maricela 776
Brown, Ross 39
Buccafurri, Francesco 639
Buyl, Ronald 657
- Cali, Andrea 473
Carmona, Josep 3, 345
Caron, Clément 130
Chakhar, Salem 784
Cherkaoui, Omar 313
Colman, Alan 258
Comi, Antonello 277
Comuzzi, Marco 166
Constantin, Camelia 130
Conti, Marco 400
Conti, Riccardo 612
- De Antonellis, Valeria 364
Debole, Franca 768
de Leoni, Massimiliano 3
De Salve, Andrea 400
De Smedt, Johannes 446
De Weerdt, Jochen 446
Diamantini, Claudia 148, 727
Di Ciccio, Claudio 75
Dirix, Michel 382
Dong, Zihe 202
Dou, Dejing 562
Dumas, Marlon 436
- Elayeb, Bilel 745, 792
Fahland, Dirk 237
Fallon, Liam 718
- Fernández, Pablo 295
Fickas, Stephen 562
Fotia, Lidia 277
Frika, Mounir 313
Frosini, Riccardo 473
- García, Luis A. 418
Gasca, Rafael M. 327
Gavriliis, Dimitris 768
Genga, Laura 148
Giannopoulos, Giorgos 553
Giroux, Patrick 130
Gmati, Fatma Ezzahra 784
Gómez-López, María Teresa 327
Gómez Sáez, Santiago 93
Griffiths, Gina 562
Gromov, Vladimir 237
Guidi, Barbara 400
Gutierrez, Fernando 562
Gutiérrez, Antonio M. 295
- Hadj-Alouane, Nejib Ben 112
Hahn, Michael 93
Han, Jun 258
Holubová, Irena 535
Huang, Haowei 202
- Islam, Md. Saiful 258
- Jílková, Eva 535
Jog, Chinmay 623
- Kabicher-Fuchs, Sonja 39
Karagiannakis, Nikos 553
Karastoyanova, Dimka 93
Kayes, A.S.M. 258
Keeney, John 718
Khoury, Selma 760
Klai, Kais 57
Kriglstein, Simone 39
- Lahbib, Wiem 745, 792
La Rosa, Marcello 436
Lax, Gianluca 639
Leggio, Daniel 674

- Lenzerini, Maurizio 580
 Le Pallec, Xavier 382
 Lepore, Lorenzo 580
 Luna, Alejandro 457
 Maggi, Fabrizio Maria 75, 436
 Mammar, Amel 112
 Maroulis, Thomas 553
 Marra, Giuseppe 674
 Martín-Díaz, Octavio 295
 Marzini, Emanuel 612
 Matteucci, Ilaria 612
 Mecella, Massimo 75
 Meghini, Carlo 768
 Melchiori, Michele 364
 Mendling, Jan 75
 Messina, Fabrizio 277
 Mileo, Alessandra 491
 Mitschang, Bernhard 21
 Moreno, Ginés 457
 Mori, Paolo 612
 Muller, Alexis 382
 Müller, Carlos 295
 Munoz-Gama, Jorge 3, 345
 Nguyen, Hoang 436
 Nicolazzo, Serena 639
 Nocera, Antonino 639
 Nuutila, Esko 682
 Nyssen, Marc 657
 O'Sullivan, Declan 718
 Papatheodorou, Christos 768
 Parody, Luisa 327
 Petrocchi, Marinella 612
 Pitto, Francesco 400
 Poggi, Antonella 580
 Polák, Marek 535
 Potena, Domenico 148, 727
 Poulovassilis, Alexandra 473
 Prades-Farrón, Miguel 418
 Räim, Margus 75
 Ramezani Taghiabadi, Elham 237
 Reimann, Peter 21
 Resinas, Manuel 295
 Reyes, Alejandro 776
 Ricci, Laura 400
 Rinderle-Ma, Stefanie 39, 327
 Rinne, Mikko 682
 Rodríguez, José 776
 Rosaci, Domenico 277
 Ruiz-Cortés, Antonio 295
 Saleh, Omran 700
 Santanchè, André 130
 Sarnè, Giuseppe M.L. 277
 Sattler, Kai-Uwe 700
 Schwarz, Holger 21
 Skoutas, Dimitrios 553
 Spognardi, Angelo 612
 Srinivasa, Srinath 623
 Storti, Emanuele 148, 727
 Suriadi, Suriadi 436
 Szabó, Ildikó 597
 Tabbane, Sami 313
 Tata, Samir 57
 Tomás, Vicente R. 418
 Ursino, Domenico 674
 vanden Broucke, Seppe K.L.M. 345
 van der Aalst, Wil M.P. 3, 237
 Van Laere, Sven 657
 Vanthienen, Jan 345, 446
 Varga, Krisztián 597
 Wang, Jianmin 184, 202, 220
 Wang, Shuhao 184
 Wang, Yuquan 220
 Wang, Zixuan 184
 Weiß, Andreas 93
 Wen, Lijie 184, 202, 220
 Wood, Peter T. 473
 Yacoubi-Ayadi, Nadia 784
 Yan, Zhiqiang 220
 Yangui, Sami 57
 Zemni, Mohamed Anis 112
 Zhang, Haotian 727